Actuation and Localization of Resource-Constrained Autonomous Microrobotic Systems

by

Brian Gregory Kilberg

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Spring 2020

Actuation and Localization of Resource-Constrained Autonomous Microrobotic Systems

Abstract

Actuation and Localization of Resource-Constrained Autonomous Microrobotic Systems

by

Brian Gregory Kilberg

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Kristofer S.J. Pister, Chair

With recent advances in miniaturized wireless system-on-chips, microactuators, sensors, and power supplies, autonomous microrobotic systems that can independently explore spaces and process information are approaching fruition. These systems, which can be smaller than 1 cubic centimeter, require actuation and localization modalities that are compatible with these strict payload and size requirements. The first part of this dissertation focuses on the development of millimeter-scale aerodynamic control surfaces actuated by electrostatic inchworm motors, intended for controlling pencil-size microrockets. The design and assembly process of these MEMS control surfaces is detailed, and they are demonstrated to produce 0.48 mN of aerodynamic lift force and induce a roll maneuver on a miniature rocket system.

The second part of this dissertation focuses on the localization of microrobotic systems using the Single Chip Micro Mote (SCμM) and HTC Vive lighthouse localization beacons. The 5 mg SCμM is a 2 mm x 3 mm x 0.3 mm wireless system-on-chip that is a suitable control, computation, and communication platform for microrobotic systems. Its integrated optical receiver, originally designed to enable contact-free programming, can be used to detect structured infrared light pulses generated by HTC Vive lighthouse stations, which are commonly used for tracking users in virtual reality systems. With this light pulse information, the SCμM system can then localize itself relative to the lighthouse base stations. The development and performance of this system, which can provide localization accuracy of up to 3.1 cm, is discussed in this part of the dissertation. Continuing work on utilizing sensor fusion algorithms to incorporate inertial measurement unit (IMU) data into this localization system is also discussed. Finally, cooperative lighthouse localization methods are discussed for use in teams of resource-constrained robots.

To my friends, family, and all my little bonsai trees

Contents

Contents **ii**

List of Figures **v**

List of Tables **xvii**

Introduction **2**

I Actuation of Miniature Autonomous Rockets **4**

1 Microrocketry **5**
1.1 Microrocketry System Requirements 6
1.2 Inchworm motors as an Actuation Modality 10
1.3 Power Supply 12

2 Inchworm Motor Control Surfaces **14**
2.1 Analysis and Design of MEMS Control Surface 14
2.2 Preliminary MEMS Control Surface Design V1 16
2.3 MEMS Control Surface V2: Improving Robustness 17
2.4 Force Characterization Experimental Setup 18
2.5 Experimental Results 21
2.6 MEMS Control Surface V3: Designing for Assembly 22
2.7 Theory and Design 29
2.8 Aerodynamic Performance 33
2.9 Systems Integration 35
2.10 Towards Full System Integration 36

3 Integrated Rocket with Control Surface **39**
3.1 Intermediate Rocket System 39
3.2 Enabling the Integrated Rocket System 40
3.3 Theory and Design 41

3.4 Integrated Rocket Design 44
3.5 Aerodynamic Performance 46
3.6 Future Vision 47
3.7 Conclusion 49

II Lighthouse Localization for Autonomous Microrobotic Systems 50

4 Lighthouse Localization of Single Chip Micro Mote 51
4.1 Introduction 51
4.2 The Single Chip Micro Mote (SCμM) 52
4.3 Lighthouse Localization 54
4.4 Single Lighthouse Validation Experiments 57
4.5 Lighthouse Triangulation 59
4.6 Results 64
4.7 Localization Improvements 64
4.8 Conclusion 66

5 SCμM Lighthouse and IMU Sensor Fusion 69
5.1 Sensor Fusion Advantages 69
5.2 IMU and Lighthouse Implementation on SCμM 69
5.3 State Estimation and Sensor Fusion Algorithms 72
5.4 State Estimation for SCμM 76
5.5 Performance with Ideal Attitude Estimation 79
5.6 Outlier Rejection with Mahalanobis Distance 82
5.7 Integrating Attitude Estimation 82

6 Cooperative Lighthouse Localization for Resource-Constrained Robots 86
6.1 Introduction 86
6.2 Related Work 86
6.3 Robotic Lighthouse Localization System 90
6.4 Wireless Reliability 94
6.5 State Estimation 95
6.6 Experimental Results 99
6.7 Wireless Sensor Node Localization Simulation 99
6.8 Conclusion 101

III Appendix 105

A Practical Assembly and Design Tips 106
A.1 Failure Modes 106

A.2 Control Surface Assembly Process 108
A.3 Silver Epoxy Caveats 109

B Scum Lighthouse Implementation Documentation and User Guide 111
B.1 Hardware Configuration 111
B.2 Interrupt Implementation 114
B.3 Pulse Type Classification 114
B.4 Pulse Processing State Machine 116
B.5 Enabling QX Series Boards to Boot 116
B.6 Optical Programming 117
B.7 Radio Calibration 119

C Lighthouse RF Synchronization 122

Bibliography 127

List of Figures

1.1 Hypothetical microrocket system. The Single-Chip μMote (SCμM) represents the SoC used for computation and communication. Zappy is the power supply, which utilizes solar cells to generate low voltage power rails for the CMOS components and high-voltage level-shifters for powering the actuators. 6
1.2 Left: Linear acceleration data from an uncontrolled 20-cm rocket launch. Right: 20-cm rocket launch trajectories reconstructed from IMU data. During launch 1, we visually measured a max altitude of 60 m, compared to the 45 m altitude derived from the reconstructed trajectory. This discrepancy is likely due to accelerometer saturation during liftoff when the acceleration was above the 16g range of the IMU. 9
1.3 Electrostatic gap closing actuator diagram. 11
1.4 Diagram of inchworm motor operation. (a.) Inchworm motor at rest with both sets of GCAs grounded. (b.) One pair of GCAs actuates, contacting and pushing forward the inchworm shuttle. (c.) The second pair of GCAs actuate, which will allow the first pair to release without the shuttle losing ground. (d.) The first pair of GCAs release which allows the second pair of motors to push the shuttle forward. (e.) The first pair of GCAs re-engage, and the processes (b-d) repeat, pushing the shuttle forward. 13

2.1 Simulated trajectory of a millimeter-scale rocket powered by a 5 mN thrust and controlled by MEMS control surfaces. The simulated rocket travelled almost 400 m at a speed of 20 m/s and demonstrated a turning radius of approximately 20 m. 16
2.2 Layout view of the bottom half of the MEMS control surface chiplet showing inchworm motors in red and the rotary slot mechanism in blue. The top inchworm motor is symmetric with the bottom one. The control surface chiplet is 7 x 9 mm^2. 17
2.3 SOI fabrication process. (a.) SOI wafer before processing. (b.) SOI wafer after evaporation deposition of 200 nm aluminum pads. (c.) Deep reactive ion etch (DRIE) patterns inchworm motors and pin joints in device layer silicon and a backside DRIE etches trenches in the silicon substrate (c.) Hydroflouric acid vapor etch releases device layer structures from the oxide layer. 18

2.4 (a.) Layout of the first iteration of the MEMS control surface design. This iteration was not functional due to a weak fin slot and inchworm pawl jamming. (b.) The fin slot was too weak to survive assembly and the insertion of the fin into the slot. 19
2.5 Image of inchworm motor shuttle jamming on pawls. This set of inchworm motors is unpowered and should be allowing the shuttle to move freely past its pawls. 19
2.6 Layout of the second version of the MEMS control surface. 20
2.7 Comparison of insufficient shuttle guides from Version 1 (Top) and sufficient shuttle guides from Version 2 (Bottom). The design from Version 1 did not have enough guides near the support spring, so the shuttle had enough freedom to move laterally as well as longitudinally. The second design version thickened the fin slot and added shuttle guides to prevent the inchworm shuttle from catching on unpowered inchworm motor pawls. 20
2.8 Fully assembled and wirebonded fin. The total size of this assembly is 7 mm x 9 mm x 3 mm. 21
2.9 Top: Rotary slot mechanism at no deflection. Bottom: Mechanism at maximum deflection of 10°. This motor was being driven at 90V and a frequency of 100Hz. 22
2.10 Top: Integrated force gauge experiment used for estimating output force of control surface. Bottom: Estimated force vs. voltage plot for the MEMS control surface. We optically measured the displacement of the force gauge and calculated the applied force by using the simulated spring constant of the structure. 23
2.11 Demonstration of fully assembled fin actuating. Top: Fin at rest. Bottom: Fin at a deflection of 5°. 24
2.12 Left: Force gauge tip placed underneath the fin. This needle was oriented vertically and the fin would rotate down into it. As the fin actuated downward, it pressed down on the needle, and the resulting force was measured by a scale below. Right: Illustration of needle-scale force gauge. 25
2.13 Measured force vs. voltage plot of the fully-assembled control surface. The inchworm motors were operating at a frequency of 500 Hz when this data was taken. 26
2.14 Comparison of fin insertion methods. (a.) Top-down fin insertion, where the fin is inserted perpendicularly into the fin slot. (b.) Side insertion, where the fin is slid into the slot from the side and is somewhat self-aligned by a funnel structure in the device layer (not shown). 27
2.15 Fully assembled MEMS control surface 27
2.16 Plane stability solutions 28

2.17 The MEMS control surface consists of an inchworm motor, a 2x4 mm^2 SOI device layer silicon airfoil, a rotary airfoil slot, and a piezoresistive half-bridge strain sensor. Six wire bonds are used to electrically connect the strain sensor and the motor. The silicon airfoil is inserted into the rotary airfoil slot and stands vertically out of plane from the rest of the device. The strain sensor measures generated lift force. 30
2.18 The inchworm motor linearly actuates the rotary joint mechanism which rotates the airfoil slot. The inset shows a microscopic photograph of the airfoil slot after rotation. 31
2.19 A patterned backside trench underneath the airfoil slot and a device-layer alignment funnel comprise the airfoil alignment mechanism. This mechanism guides the vertically-oriented airfoil into this slot as it is inserted. During airfoil insertion, cantilever springs align and secure the airfoil. 32
2.20 The strain sensor consists of a substrate layer cantilever with device layer piezoresistors patterned on top of the cantilever. When the cantilever deflects, it induces a differential strain in the piezoresistors on the edges of the cantilever. The anchor of the strain sensor is glued to a mount, so the rest of the device is suspended. When the airfoil rotates in airflow, it generates an aerodynamic lift force that deflects the strain sensor. 33
2.21 Top: Measured relationship between lift force, angle of attack, and airspeed for a 2x4 mm2 silicon airfoil. In left graph, experiments were run at 15 m/s flow speed. Bottom: a device with an immobilized airfoil in a wind tunnel was rotated while measuring the lift force. 34
2.22 Image of the MEMS control surface system mounted for aerodynamic force experiments. 34
2.23 Aerodynamic performance graph. Top: Lift force generated by a periodically actuated airfoil. Noticeable force spikes occur when the fin is rotated by 11 degrees. Bottom: Control experiment showing force exerted on strain sensor when no actuation is occurring in 23 m/s airflow. In this control, the motor is running but is decoupled from the airfoil, so the airfoil doesn't deflect. In both plots, the 4 second moving average was subtracted from the signal . . . 35
2.24 Integrated rocket system components. Top: Flexible PCB designed for assembling and routing control and power signals for the control surface system. The PCB is wrapped around a 3D-printed MEMS control surface fuselage module. Bottom: All of the electronics necessary to utilize the MEMS control surfaces has been demonstrated in a 1.5 cm diameter body tube. 36
2.25 Pencil-sized rocket with one assembled MEMS control surface, LiPo battery, and control electronics attached 37
2.26 Cross-sections of an island buckle joint illustrate how it prevents out-of-plane dislocations of the device-layer mechanisms. Movement in the positive z direction is constrained by chiplet-anchored cantilevers, while movement in the negative z direction is contained by island-anchored cantilevers. 37

3.1 Annotated image of MEMS control surface flex PCB. Four control surfaces can be mounted onto this board at onfcce. This flex PCB has stiffener underneath the wirebond and ground pads, which keeps the wirebond surface and substrate bonding surface flat. Manufacturer-standard flex stiffener could also be used as a stiffener for these pads. The header pins are soldered onto the connection pads and then are encased in epoxy to increase their structural strength. . . . 41
3.2 Example of an assembled control surface PCB mounted on a rocket fuselage module. 42
3.3 The fully assembled rocket, consisting of a MEMS control surface and its 2x4 mm2 steel airfoil, a single-cell LiPo battery, and control electronics. 42
3.4 (Left) Schematic of half of the MEMS actuator mechanism. The inchworm motor linearly actuates the rotary pin joints, which rotate the airfoil slot. (Right) A microscopic photograph of the airfoil slot after rotation. 44
3.5 The assembled MEMS control surface. 45
3.6 (Left) The 13 mm diameter high voltage buffer board. (Right) The MIMSY control and wireless node. 46
3.7 Block Diagram of Rocket Electronics System 46
3.8 Roll angle vs. time data for individual rocket experiments in the wind tunnel, measured visually. The plots show the difference between the rocket's roll with and without an actuated fin. From left to right, the experiments were conducted at $K_{torsion} = 2.6\mu$Nm, 4.9μNm, and 0.51μNm. The $5.1°$ and $10.5°$ experiments were conducted in 13.3 m/s airflow while the $19.6°$ experiment was conducted in 20.1 m/s airflow, which explains the latter's large roll angles. 47
3.9 (Left) Lift force is generated when the fin is rotated, causing the entire rocket to roll along its axis. (Right) Schematic of the integrated rocket system inside a wind tunnel. 48
3.10 Output torque from the fin's lift force, measured across 3 trials for each angle of attack. All measurements were normalized to match a 2x4 mm^2 airfoil's area and a wind speed of 13.3 m/s. The rockets were either for 1. electronically actuated or 2. statically positioned systems. 48
3.11 Current and projected rocket volumetric dimensions and mass with future scaling of rocket components. The control surfaces presented in this paper are well suited for miniature rockets as is, and incorporating the referenced circuitry and rocket engine designs could further reduce rocket size. The total row includes the above factors plus the rocket's nose cone, electrical wiring, and fuselage. 49

4.1 Die photo of the Single Chip Micro Mote (SCμM). The integrated optical receiver is initially designed for contact-free optical programming. In this paper, we re-purpose it to allow the mote to localize itself by timing the horizontal and vertical laser scans from a HTC Vive lighthouse localization system. . . 52

4.2 Triangulation of lighthouse measurement rays. The relative angle measurements from each lighthouse create rays pointing in the direction of the mote. These projected rays are unlikely to cross due to measurement noise, so the direct linear transform is used to estimate their intersection and triangulate the SCμM's position. 53
4.3 SCμM receiver summary [65]. 54
4.4 The optical programming process of SCμM (here shown soldered onto its development board). An optical programming board, connected to a laptop, blinks an LED: it starts by a bootload blink pattern, followed by the stream of bits contained in the binary image to load. Upon receiving the bootload blink pattern, dedicated circuitry in SCμM copies the subsequent bit stream into memory, then resets the micro-controller. The optical receiver is repurposed to time the flashes received from a lighthouse base station during the execution of the firmware loaded, to allow the SCμM chip to self-localize. 54
4.5 Illustration of lighthouse localization as implemented in the HTC Vive V1. a.) The system consists of a lighthouse beacon containing a wide-angle LED array, two sets of rotating planar lasers, and a photodetecting system. b.) As the the rotating laser scan passes a reference heading, the LED array flashes and generates a synchronization pulse, which is detected by the photodetector. c.) Once the planar laser scan reaches the photodetector, the photodetector detects the scan and measures the time difference between the sync pulse and the laser scan. d.) Using the constant rotation frequency of the laser scan and the time difference, the photodetector can calculate its heading relative to the reference heading of the lighthouse beacon. 56
4.6 Chronogram of the pulses received by the SCμM chip from the two lighthouses. The lighthouses are synchronized over a wired interface. They alternate between periods when sending omnidirectional sync pulses, and periods when sending laser pulses. Lighthouse 1 always sends a sync pulse before Lighthouse 2. The duration of the sync pulse indicates what the lighthouse will do, per Table 4.1. The duration before receiving the laser pulse encodes the angle to the lighthouse. In this illustration, SCμM measures 120 degrees azimuth / 70 degrees elevation from Lighthouse 1, 110 degrees azimuth / 150 degrees elevation from Lighthouse 2. 57
4.7 Illustration of lighthouse experimental setup. The blue line indicates the ground truth 3D trajectory of the optical receiver, obtained by motion capture. The red dot indicates the location of the lighthouse beacon, and the field-of-view indicates the direction the lighthouse beacon is pointing. 58
4.8 Azimuth and elevation tracking of optical receiver, compared to ground truth. The RMS error of azimuth tracking was 0.386 degrees. Elevation tracking's RMS error was 0.312 degrees. 59
4.9 Performance of azimuth and elevation tracked by the optical receiver compared to the ground truth trajectory. 60

4.10 Camera model of a lighthouse base station. The azimuth and elevation measurements are projected onto the virtual unit distance image plane of the lighthouse. 62
4.11 Illustration of lighthouse calibration. The calibration process uses the projected measurements and a small portion of the OptiTrack truth trajectory data (this data was not used to calculate measurement accuracy). The PnP algorithm, using the iterative nonlinear least squares Levenberg-Marquardt method, generates the optimal camera projection matrix given this input data. This plot compares the projected lighthouse image points to the ground truth points projected to the lighthouse image plane using the calibrated projection matrix for Lighthouse 1. 63
4.12 Experimental azimuth and elevation measurements of a SCμM mote compared to motion capture ground truth. The lighthouse-relative ground truth azimuth and elevation angles were determined by using the calibrated lighthouse poses to project the 3-dimensional world-frame ground truth trajectory to 2-dimensional lighthouse-relative azimuth and elevation angles. 65
4.13 Violin plot showing distribution of azimuth and elevation angle error. One outlying data point that is 80 degrees from the mean of the elevation B data is not shown. Data points outside of these percentiles are not shown. The majority of error points are within a tight distribution, with a few outliers far from the mean. The RMS tracking error, with outliers (error >10 degrees) removed, is 0.63 deg, 0.37 deg, 0.60 deg, and 3.87 deg for azimuth 1, elevation 1, azimuth 2, and elevation 2, respectively. 66
4.14 (a) Triangulation of lighthouse data to reconstruct the trajectory of the SCμM chip throughout the experiment. Large deviations in tracking are caused by missed measurements. In our system, a missed measurement gets replaced by the latest measurement. This allows for the system to always display a position, but causes the reconstruction algorithm to use out-of-date measurements, resulting in inaccurate positioning. The subset of data used for lighthouse calibration is excluded from this data. (b) Triangulation error of lighthouse tracking over time. The RMS tracking error has significant temporal deviation throughout the experiment, characterized by periods of time with minimal error and periods of time with significant error. Periods of significant error are likely correlated with line-of-sight occlusion, infrared interference, or RF interference events. 67
4.15 Percent of lighthouse measurements received, as a function of distance to lighthouse. The reliability of the lighthouse measurements drops off at around 1 m. An active motion capture system is used in this experiment to measure SCμM position, which could have a small effect on the lighthouse reliability due to infrared interference. 68

4.16 Effect of incidence angle on reliability of lighthouse measurements. An angle of 0° corresponds to the face of the SCμM parallel to the face of the lighthouse base station, with the laser scans perpendicular to the face of SCμM. A rapid dropoff in lighthouse measurement reliability occurs when SCμM is rotated >70° relative to the lighthouses. In this experiment, the lighthouses are 9 cm apart, the SCμM was 47 cm away, centered in between the two lighthouses. The incidence angles are measured manually, rather than by a motion capture system. 68

5.1 Selected advantages of IMU sensor fusion for improving lighthouse localization on SCμM. Sensor fusion tracking systems can update at the same rate as the IMU and can provide a tracking estimate even when lighthouse measurements are occluded or aren't available. Nonlinear estimation enabled by sensor fusion can also avoid issues that arise when triangulating using multiple asynchronous measurements that may be delayed. Nonlinear estimation allows the integration of individual lighthouse angle measurements, rather than requiring four asynchronous angle measurements to generate a triangulated measurement. 70

5.2 Statistical outlier rejection using Mahalanobis distance [76]. This estimator is using SCμM lighthouse data with IMU data. State estimation techniques used in sensor fusion keep a continuous estimate the probability distribution of the SCμM's position and state. Using this probability distribution and a model providing predicted lighthouse measurements as a function of SCμM state, sensor fusion algorithms can determine the likelihood of a measurement being an outlier. The Mahalanobis distance is one measure of this, which is roughly the number of standard deviations a measurement is from its predicted measurement, given SCμM's current state. 71

5.3 Lighthouse and IMU data gathered by SCμM during a brief experiment. At constant z position, azimuth measurements are proportional to x position and elevation measurements are proportional to y position, given that $\frac{x}{z} = tan(\theta_{az})$ and $\frac{y}{z} = tan(\theta_{el})$. As a result, the second derivative of these angles is roughly equivalent to the IMU accelerations measured on those axes. The IMU data rate was 50 Hz in this experiment. The x axis is 10 Mhz timer ticks, so each tick corresponds to 100 μs. 72

5.4 Illustration of Bayesian state estimation 74

5.4 Illustration of Bayesian state estimation 75

5.5 Decoupled architecture vs. fully-coupled architecture. 76

5.6 Proposed cascaded filter architecture for lighthouse sensor fusion. 77

5.7 Comparison of linear kalman filter architecture to UKF architecture. 79

5.8 Architecture used to evaluate position estimation, independent of attitude estimation. Accelerometer and gyroscope measurements were derived from the ground truth pose data from the experiment. Attitude was taken directly from the ground truth data in order to isolate the performance of the position estimator. 80
5.9 Tracking comparison between the linear and nonlinear state estimators. The UKF improves performance in areas where some measurements lose reliability do to occlusion or lost measurements, such as around 105 seconds in the plots above. 80
5.10 Selected example showing the benefits of nonlinear estimation. The linear Kalman filter is limited to using the triangulated lighthouse measurements, which in this case are inaccurate due to outdated angle measurements being used in the triangulation function. There are still some lighthouse measurements being received, but all four of the necessary angle measurements aren't being updated, so the triangulation algorithm uses the most recent measurements. The UKF can handle this error case, since it can use each individual measurement to update, rather than needing four measurements that are asynchronous. 81
5.11 Sensitivity of UKF to process covariance. While the UKF provides improvements over the Kalman filter for lighthouse localization, it is more sensitive to calibrated parameters such as the process and measurement covariances, and it can diverge at certain parameters. This can increase the difficulty of UKF implementation compared to the Kalman filter, which doesn't experience this issue. 81
5.12 Outlier rejection using Mahalanobis thresholding. The threshold used to reject oultiers in this experiement was 5. 83
5.13 Position tracking results from sensor fusion experiment. The bottom plot shows the Mahalanobis distance of each measurement the estimator's predicted measurement distribution. Outlier measurements with a Mahalanobis distance greater than 2.5 were ignored by the tracking algorithm, the results of which can be illustrated by the lighthouse measurements plotted in the position graphs. 84
5.14 Attitude tracking performed by Madgwick filter. Pitch and roll euler angles were successfully estimated without drift by the Madgwick filter. The filter could not provide a drift-free estimate of the yaw euler angle, which is about the z axis and the gravity vector. Providing magnetometer data to the Madgwick filter would fix this issue. 85

6.1 Microrobotic lighthouse setup. Microrobots, such as the hexapod [7] or ionocraft [5] shown in this figure, could act as lighthouse beacons by rotating while equipped with laser diodes and line-generating optics. In this system, the microrobots would have to keep track of their heading as they rotate, and periodically transmit it to other robots that are using the beacon laser to localize. 87
6.2 Schematic of proposed distributed lighthouse self-localization. The quadrotor can acts as a lighthouse base station by rotating in order to provide and obtain relative headings distances between itself and wireless sensor nodes. In this system, the quadrotor can localize itself based on bearing measurements relative to anchor sensor nodes with node locations, and it can provide bearing measurements to sensor nodes with unknown locations, which allows them to localize themselves 88
6.3 Crazyflie minidrone platform with a MIMSY, 850nm laser, plane-generating optics, and an optical flow deck 89
6.4 a). The lighthouse robot rotates while recording orientation-timestamp information from its attitude estimator. b). Sensor 1 detects laser sweep using its photodetector and records the network-synchronized timestamp of the laser detection event at T_{node}. c). Lighthouse robot periodically broadcasts timestamp-orientation mapping, which other robots use to localize themselves. d). Sensor 1 uses the received time-orientation broadcast and T_{node} to calculate its relative heading θ_{node}. e). Sensor 1 sends the relative heading back to the lighthouse robot. 91
6.5 System Block Diagram 92
6.6 Calculation of heading from laser pulse timing. Once a node detects a laser pulse and receives a localization EB, it can calculate its heading relative to the lighthouse node by interpolating the surrounding lighthouse time-heading tuples and looking up the laser pulse time. This data was obtained during an experiment with a fixed-position lighthouse mote rotating and a stationary self-localizing mote. This heading data was produced with the onboard magnetometer on a MIMSY that was located indoors. 93
6.7 OpenWSN motes periodically broadcast packets called enhanced beacons (EBs), which contain network-related data. EBs contain Information Elements (IEs), which are containers used to store specific types of payload. The lighthouse mote uses EBs to broadcast time-heading tuples. 94

6.8 Time-synchronized channel hopping (TSCH) matrix for OpenWSN. Timeslots in the TSCH matrix correspond to points in time and frequency and can be scheduled as active and inactive. Active slots can be used for transmitting or receiving packets, while inactive slots are used for saving power. All motes in a TSCH network must maintain a common synchronized sense of time in order for this network architecture to operate. The absolute slot number (ASN) reflects the current network-wide time in units of timeslot, which are 10 ms long in this application. This universal timebase enables infrastructure-free lighthouse localization. 96
6.9 Reliability of robot to anchor beacon transmissions within OpenWSN. Each broadcast from robot to anchor consisted of three redundant retransmissions of the broadcast's packet. Packet delivery rate (PDR) was calculated as the ratio of all packets (including redundant retransmissions) that are successfully received by the anchor, which is also approximately the success rate of a broadcast using not using redundant transmissions. Transmission success rate was the percent of broadcasts where at least one of the three retransmitted packets were received. RF interference within the experiment space decreased the packet, which is evident in the low PDRs at short distances. Retransmitting packets three times improved the success rate of the transmissions. 97
6.10 X, Y, and Z state estimates during a quadrotor lighthouse experiment. Initial lighthouse bearing measurements were heavily weighted due to high initial state variance (see Figure 6.11). This is illustrated by the large discontinuities in tracking between 165 and 175 seconds, which correspond to lighthouse measurement updates. As the Crazyflie received more measurements from the sensor nodes, the state variance decreased (Figure 6.11), the measurements were less heavily weighted, and the estimator started to recover and track with less error. The bottom graph shows the received lighthouse measurements. The Z state estimation was very accurate and can be attributed to Bitcraze's stock state estimation with optical flow deck, which includes an infrared range sensor. The bottom graph shows the quadrotor MIMSY lighthouse measurements (MLH) received by the quadrotor. 98
6.11 Estimator Variance throughout the experiment. The MIMSY lighthouse measurements (MLH) decreased the estimated variance of the x and y states, as expected 100
6.12 Estimation error during the quadrotor lighthouse localization experiment. The quadrotor is performing rotations and receiving bearing measurements from sensor nodes. 101
6.13 Pose estimation during experiment. Pose was estimated using the stock Crazyflie EKF algorithm, which relies on gyroscope measurement data. . . . 102

6.14 Measurement error during quadrotor lighthouse experiment. Top: bearing measurements compared to actual relative bearings determined by ground truth data. Bottom: measurement error. The error bias (mean error) was -0.341 radians, and the standard deviation was 0.431 radians 103
6.15 Simulation of wireless sensor node localization using lighthouse bearing measurements from a rotating quadrotor that is also moving translationally. In this simulation, the sensor node doesn't know its location and is estimating its location from bearing measurements from the quadrotor lighthouse. As the anchor sensor mote receives more measurements, its location estimate improves and estimated variance decreases. 104

A.1 Example of silver epoxy reflow. Silver epoxy residue can be see surrounding the outline of large portions of the silicon grid, even though the epoxy was only applied on one point of the grid. This wicking is exacerbated when the device is heated to cure the epoxy. 107
A.2 Out-of-plane protection bracket. Silver epoxy is applied to the edges of the device sustrate, rather than on top of the device layer. This reduces the risk of silver epoxy reflowing and affecting the MEMS devices. 110

B.1 Block diagram of GPIO configuration required for lighthouse localization implementation on SCμM. GPIOS 1, 8, and 9 must be electrically connected. 112
B.2 Illustration of lighthouse pulse detection using level-sensitive interrupts. . . . 115
B.3 State machine for processing lighthouse pulses containing azimuth measurements. The pulse processing for elevation measurements is identical to elevation, so its state machine is not shown. During state transitions that are triggered by receiving an azimuth sync pulse (AZ), the state machine records the time of the pulse's rising edge. During laser sweep pulse (LASER) transitions, the state machine will transmit a packet containing the time of the sync pulse's rising edge and the time of the laser pulse's rising edge, which can be used to calculate the SCμM's azimuth angle relative to the lighthouse base station. State transitions with "/*" execute the action "*" during the transition. State transitions without "/*" do not have any accompanying actions. 117
B.4 A QX series board with LDO workaround. The 1.2 V LDO is powered by the board's VBAT rail and provides a 1.2 V output to SCμM's VDDD connection. 118

C.1 Range comparison between laser sweeps and omnidirectional synchronization pulses. The range was determined by increasing the distance between the photodiode receiver and the lighthouse beacon until the the light pulses could no longer be seen by the receiver. 122

C.2 Timing of Lighthouse OpenWSN Integration. OpenWSN uses the protocol 6TiSCH protocol [121] to implement time-synchronized channel hopping. In this case, the minimal 6TiSCH cell is the minimal required active slot in the schedule, and determines when nodes in the network can communicate with each other. 123
C.3 Broadcast of Enhanced Beacons (EBs) containing Information Elements (IEs) with period and timing information of lighthouse sync pulses. 124
C.4 Timing of Lighthouse OpenWSN Integration. The sync pulse time of arrival is captured in the network-synchronized time base, which is the time of the current ASN plus the timer measured offset between the the time the pulse occurred and the time corresponding to the beginning of the current timeslot. 124
C.5 Illustration of lighthouse sync pulse drift. Nodes in the network periodically receive the exact timing of synchronization pulses from the sync pulse detection node. Between these resynchronization packets, motes predict when the next sync pulses occur based on their period. The motes can then use these predicted pulses to localize when they receive laser sweeps. As the amount of time since the last sync pulse reference packet was sent increases, the projected pulse times will drift and accumulate error. 125
C.6 Left: Illustration of drift between sync pulse projection and actual sync pulse. Right: Measured frequency of lighthouse sync pulses. 125
C.7 Drift error of lighthouse sync pulses timing, with and without drift correction. 126
C.8 Timing error of lighthouse synchronization pulse with respect to the projected sync pulse timings. The mean error was 32 μs, the standard deviation was 95 μs, and the timing resolution was 30.52 μs. 126

List of Tables

1.1 Summary of mass and size of microrocket system components. 8

3.1 Summary of 20 cm Rocket System Components 39

4.1 Duration of the pulse activity of an HTC Vive Lighthouse. 55

5.1 Positioning Accuracy of Lighthouse-IMU Sensor Fusion of the experiment shown in figure 5.13 85

B.1 Duration of the pulse activity of an HTC Vive Lighthouse. 115

Acknowledgments

I could not have completed this long and arduous journey through graduate school without the help of numerous professors, students, friends, and family members. I'd first like to thank my advisor Kris Pister, whose enthusiasm and support motivated me through these past five years. I'm truly grateful for his willingness to let me to work on projects that excited me. I would also like to thank my Claire Tomlin, Mark Mueller, and Steve Glaser for sitting on my qualifying exam committee.

My fellow students were an integral part of my experience at Berkeley. My first mentor was Travis Massey, who has a seemingly endless amount of practical lab and engineering knowledge, and helped me solve numerous problems in the early days of my PhD. The Pister lab MEMS crew, Joey Greenspun, Daniel Contreras, and Daniel Drew, were extremely helpful in teaching me the practical aspects of MEMS robotics design. As my work shifted to the Single Chip Micro Mote, David Burnett, Fil Maksimovic, and Brad Wheeler were always willing to answer my numerous questions about the inner workings of the system and enabled much of my work using the chip. I would also like to thank my fellow Pister lab peers, Craig Schindler and Hani Gomez, who navigated with me the twists and turns of prelims, quals, and PhD life these past five years. My undergraduate students, Felipe Campos and Ahad Rauf, made invaluable contributions to my work and were a pleasure to work with and mentor.

As I forayed into the wireless sensor networking space of OpenWSN, Thomas Watteyne, Tengfei Chang, Timothy Claeys, Yasuyuki Tanaka, and Jonathon Munoz were helpful guides. I would also like to thank the EVA team at Inria for hosting me in Paris.

My friends, family, and Rachel brought joy and meaning to my life that helped me persevere through this journey, and I am forever grateful. I'd finally like to thank my roommates in the Pristine Palace, who have helped keep me sane as I wrote this dissertation in the midst of a global pandemic and shelter-in-place order.

Introduction

Autonomous Microrobotic Systems

Microrobots are highly miniaturized robotic systems that often have a volume on the order of 1 cm^3 or smaller. Their modes of mobility can include a variety forms, such as flying flapping wing robots [1, 2, 3, 4], hovering ionocrafts [5], walking robots [6, 7, 8, 9, 10], and jumping robots [11, 12, 13, 14, 15, 16]. Due to their size, autonomous microrobotic systems have limited payload and power capabilities. For example, the hovering ionocraft in Drew et al. [5] has little payload capacity beyond the 37 mg inertial measurement unit it can carry. These limitations have an important impact on the system components that are suitable for microrobotic systems.

Payload constraints affect the size and type of actuators that can be used. Many standard robotic actuators, like servos, are too large or heavy for use in microrobotic systems. This has led to extensive development of actuators suitable for microrobotic applications, with examples including piezoelectric motors [17, 18, 1], thermal actuators [19], electrothermal shape memory alloy actuators [8, 20], electromagnetic actuators [16, 10], and electrostatic motors [7, 14, 9, 13]. Electrostatic inchworm motors provide a favorable tradeoff between force and displacement, while also being suitable for batch fabrication [21], [22]. These motors are also easy to integrate into complex MEMS mechanisms. This dissertation focuses on the use of electrostatic inchworm motors to design a suitable control surface for microrockets.

The system constraints of microrobots also affects the maximum computational power and complexity they can support. This especially affects the sensing and autonomous navigation capabilities of these robots, which are important components for autonomous microrobotic systems. While computational hardware enabling sophisticated autonomous navigation abilities such a visual inertial odometry (VIO), simultaneous localization and mapping (SLAM), and computer vision has miniaturized to the point where modern high performance drone systems, including both academic [23, 24] and commercially available[1] systems, can leverage these capabilities for advanced autonomy. Unfortunately, the computation platforms that support these capabilities, such as the Qualcomm Flight and Nvidia Jetson modules, are around the size of a credit card, and are still too large and heavy for microrobots. Microbotic systems are limited to smaller and lower power microcontrollers and lightweight sensors like inertial measurement units (IMUs), optical flow sensors, low resolution cameras, and photodetectors. For example, one suitable computation platform for microrobots is the Single Chip Micro Mote (SCμM), which is a 5 mg, 2 mm x 3 mm x 0.3 mm wireless system-on-a-

[1]see Skydio 2 drone (`https://www.skydio.com/`)

chip with a Cortex M0 processor and an IEEE 802.15.4 radio with limited Bluetooth Low Energy transmit capabilities [25]. Another example of a suitable computation platform for microrobots is the Michigan Micro Mote (M3), a 2 x 2 x 4 mm^3 wireless communication system that includes a 160 x 160 pixel image sensor and energy harvesting capabilities [26]. The form factors of SCμM and M3 make them suitable for integration into an autonomous microrobotic system. Unfortunately, the reduced computation power of these platforms precludes the use of modern high performance robotic navigation systems for microrobotic systems.

This dissertation discusses the development of actuators and localization systems for autonomous microrobotic systems. The first part details the design of aerodynamic control surfaces that utilize electrostatic inchworm motors [21, 22], intended for controlling micro-rockets and other pico-air vehicles. The integration of these control surfaces into a miniature autonomous rocket system is also discussed. The second part of this dissertation covers the use of lighthouse localization, a localization system commonly used in industrial metrology [27] and virtual reality [28], to localize microrobots. SCμM includes an integrated photodiode, initially intended for optical bootloading, which can also detect the structured light emitted by a lighthouse localization system. This allows SCμM to be accurately localized by these systems. A cooperative lighthouse localization method, intended for teams of robots operating in areas where standard lighthouse localization infrastructure isn't present, is also discussed.

Part I

Actuation of Miniature Autonomous Rockets

Chapter 1

Microrocketry

Rocketry has been a highly effective technology for rapid transportation over large distances, especially in space exploration and space transportation. Currently, most useful rocketry technology is far from portable and is firmly limited to the macroscale. Increases in portability from scaling down rocketry to millimeter-scales could prove to be useful. For example, portable millimeter-scale rockets could be used to deploy large networks of wireless sensors quickly and accurately from a single host vehicle, enabling the ability to gather information from a large target area. This could be very useful for time-sensitive situations like search and rescue, and military operations.

Militaries are increasingly concerned about swarm attacks from low-cost micro air vehicles [29], which will lead to a need for smaller cheaper anti-MAV missiles, perhaps stopping short of the nano-scale predicted by Stephenson [30]. These rockets would benefit from recent improvements in battery energy density and the decreasing size, power, and cost of digital computation and sensors, which could enable low-cost, highly maneuverable autonomous MAVs with size scales $<$15 cm.

One of the smallest guided rockets currently developed is the US Navy's 64 cm long, 57 mm diameter Spike missile. Miniaturizing rocket guidance systems even further will require reducing the size of rocket propellant systems, control and communication electronics, and control surface mechanisms. Previous research has already miniaturized composite propellant rocket motors [31, 32, 33] and developed 2x3 mm^2 wireless system-on-chips that can be used as miniaturized avionics and telemetry platforms [25]. These technologies are developed enough that they could be feasibly integrated into a microrocket in their current state.

The benefits of miniaturization are not limited to rockets; small unmanned aerial systems (sUAS) and micro-air-vehicles (MAVs) have demonstrated useful applications such as search and rescue, aerial photography, crop inspection and biopsy, and industrial chimney inspection [34, 35, 36]. Scaling these systems down to the pico-air-vehicle (PAV) realm, where dimensions and masses less are than 5 cm and 500 mg, will improve energy consumption, decrease cost, increase ensemble density, and increase data granularity of unmanned aerial systems [36].

1.1 Microrocketry System Requirements[1]

Microrockets will require multiple technological innovations to operate feasibly. Among these innovations are miniature form factor computation, low power and lossy communication systems, miniaturized rocket propulsion, and miniature control surfaces. Recent advances in mesh networking, MEMS technology, and novel propulsion methods are increasing the feasibility of microrockets and pico-air-vehicles [36]. This section will enumerate the system requirements for an autonomous microrocket and explore the current state of the art.

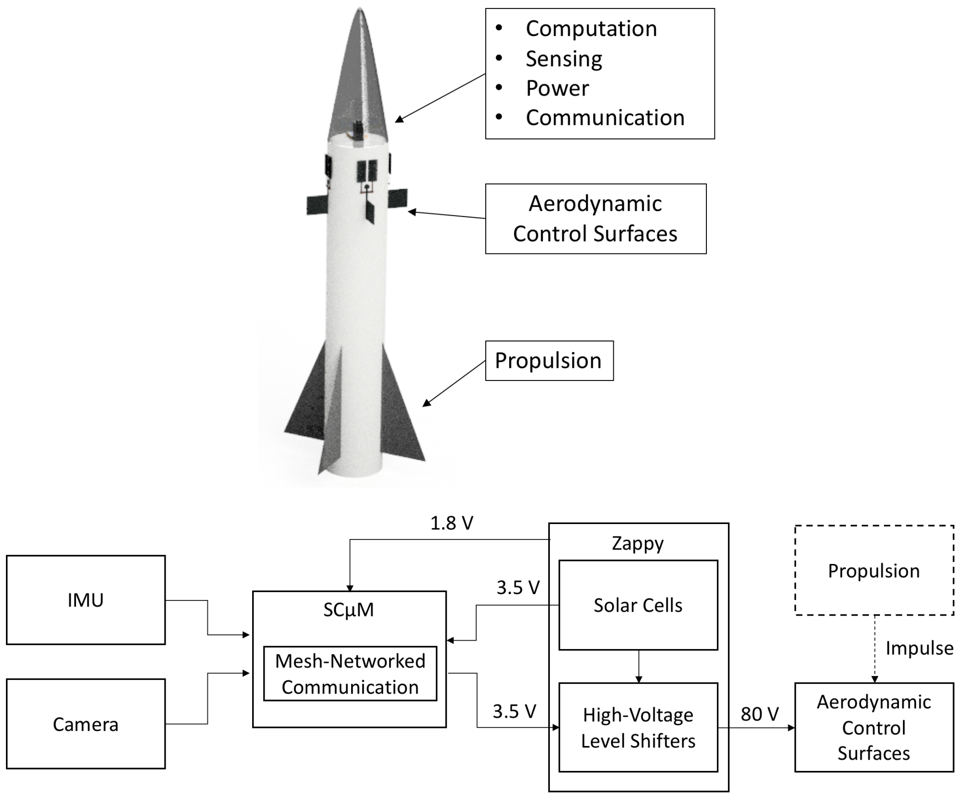

Computation
Sensing
Power
Communication
Aerodynamic Control Surfaces
Propulsion
IMU
Camera
1.8 V
SCμM
Mesh-Networked Communication
3.5 V
3.5 V
Zappy
Solar Cells
High-Voltage Level Shifters
80 V
Propulsion
Impulse
Aerodynamic Control Surfaces

Figure 1.1: Hypothetical microrocket system. The Single-Chip μMote (SCμM) represents the SoC used for computation and communication. Zappy is the power supply, which utilizes solar cells to generate low voltage power rails for the CMOS components and high-voltage level-shifters for powering the actuators.

Propulsion

Propulsion determines many of the limiting factors of a microrocket system, chiefly maximum payload, so it is important to start with propulsion when specifying the system requirements

[1]Parts of this section are adapted from [37, 38, 39]

of a microcket. Effective millimeter-scale rocket motors have already been developed. For example, Lindsay et al. demonstrated a dime-sized rocket motor that was capable of short flights [31]. Kovac et al. developed 3-6 mm diameter rocket motors with specific impulse of 27 seconds and burn times of up to 40 seconds [33]. These motors could provide a millimeter-scale rocket with up to 800 meters of range at sea level.

Available propulsion determines the maximum payload, and thus many of the system limitations a microrocket would have. Rocket motors and their propellants are commonly characterized by their impulse (Equation 1.1) or specific impulse (Equation 1.2). In these definitions, F is the amount of average amount of force a motor generates over a period of time, Δt is the time period length, $\dot{m}$ is the mass flow of the propellant, and g is gravitational acceleration. Furthermore, the motors in [33] were characterized by the construction efficiency of their combustion chambers, using the impulse per weight ratio, I_w, shown in Equation 1.3 where m_c is the mass of the combustion chamber.

$$I = F\Delta t \tag{1.1}$$

$$I_{sp} = \frac{I}{\dot{m}g} \tag{1.2}$$

$$I_w = \frac{I}{m_c g} \tag{1.3}$$

For example, the motors in [33] have an I_{sp} of 27 seconds, i.e. they can provide 0.27 Ns of impulse per gram of propellant. Therefore, a 2 gram motor (0.54 Ns impulse) with a 10 second burntime could produce 54 mN of average thrust, which equates to a maximum liftoff mass of roughly 5 grams. To achieve of take-off acceleration of 0.5 g, the takeoff mass must be under 3.5 grams, which yields a 1 gram payload mass budget, taking into account the mass of the propellant and the .5 g mass of a combustion chamber required, which have an impulse to mass ratio of 1.05 Ns/g [33]. Additionally, less mass and smaller sizes lead to large improvements in maneuverability, as will be explained later in this text, so there are clear incentives to minimize the mass and size of the rocket system. A rough mass limit of 1 grams restricts the use of many commercially-available computation, communication, actuation, and power system components.

Computation

A suitable computation platform is paramount for a controlled microrocket. A microprocessor platform enables sensing, actuator control, navigation algorithms, and communication. While centimeter-scale microcontroller platforms are widely available, even the smallest microcontroller platforms are larger than 1 cm^2. For example, the Micro-Inertial Measurement System (MIMSY) is one of the smallest wireless communication and computation platforms available and is 1.6 x 1.6 cm^2 [40] with a mass of 1.4 grams. While this is quite suitable for a small rocket, MIMSY is not suitable for the size and mass limitations of microrockets. The size and

System	**Dimensions (mm^3)**	**Mass (g)**	**Operating Voltage (V)**
Propulsion	6x6x30	2.5	N/A
Actuation	5x9x0.5	0.060	30-80
Power	3x3x0.7	0.020	1.8, 80
Computation and Communication	2x3x0.3	0.005	1.8
Inertial Sensing	3x3x1	0.010	1.8
Camera	1.5x1.5x1.5	0.010	1.8
Total	6x6x35	2.605	1.8, 80

Table 1.1: Summary of mass and size of microrocket system components.

mass of current commercial off-the-shelf (COTS) microcontroller platforms are large enough to significantly effect the moment of inertia, mass, and drag of a microrocket, rendering them non-viable. For example, the 1.4-gram MIMSY is greater than the microrocket's hypothetical payload budget of 1 gram. Furthermore, the 1.6 cm width of MIMSY is over twice as large as the 6 mm diameter of the motors from Kovac et al [33]. Clearly, a smaller platform is required to enable autnomous microrobots. Recent research has made progress on deveoloping miniaturized computation and communication platforms. For example, the Single Chip μmote (SCμM) [25] is a 2 x 3 x .3 mm^3 wireless SoC with an ARM Cortex M0 processor and an IEEE 802.15.4 and Bluetooth Low Energy (BLE) transmit compatible radio. This device only requires a power, ground, and antenna connection to operate fully. It does not require a crystal oscillator or any other passive components, which reduces the footprint and of the system. The 5 mg SCμM could easily be carried as payload by a state-of-the-art microrocket motor [33] without significantly impacting the performance of the rocket. This SoC is developed to the point that it can currently provide an interface to an inertial measurement unit for navigation, and provide communciation, and actuator control for a miniature autonomous microrocket. Furthermore, this SoC can easily operate inchworm motors, which will be used by our proposed MEMS control surfaces [41].

Communication

Communication will be helpful for both controlling microrockets and receiving data from them. Wireless communication in this context is severely limited by the size constraints of the communcation systems. Size affects the communication performance in two ways: it limits power consumption, and limits antenna effeciency, as antennas have a wavelength-dependent dimension that maximizes efficiency size, and at 2.4 Ghz this length is larger than SCμM, which reduces range considerably. Fortunately, recent developments in OpenWSN [42] and mesh networking for low power and lossy networks can allow low range and lossy radios to

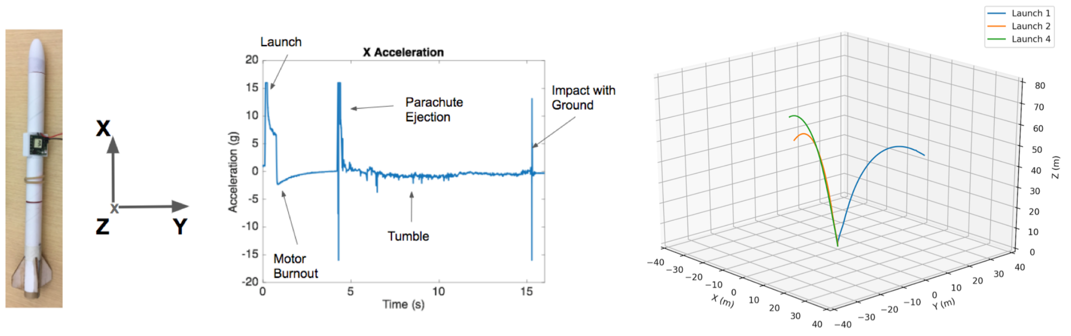


Figure 1.2: Left: Linear acceleration data from an uncontrolled 20-cm rocket launch. Right: 20-cm rocket launch trajectories reconstructed from IMU data. During launch 1, we visually measured a max altitude of 60 m, compared to the 45 m altitude derived from the reconstructed trajectory. This discrepancy is likely due to accelerometer saturation during liftoff when the acceleration was above the 16g range of the IMU.

extend their range. Mesh networking allows a node to relay its communication to a base station through multiple other nodes in range, effectively increasing the communication range as long as there are enough nodes in range to relay the message. With the high swarm density that miniaturized rockets would allow, it is feasible to rely on mesh networking to relay messages from rockets to the base station. Unfortunately, this won't work for singular rockets without the supporting infrastructure of other network nodes nearby. Wireless communication can still be used for things like target acquisition or navigation destination setting.

Sensing

Recent progress in low-cost, miniaturized inertial measurements units (IMUs) makes inertial navigation a real possibility for microrockets. For example, the ICM-20948 from TDK Invensense is 3 x 3 X 1 mm^3, which is a form factor that is appropriate for providing inertial navigation for a microrocket. This chip is on the order of 10 mg, which limits its impact on the payload budget of a microrocket system. While extended navigation with an IMU will not work due to accumulated error, IMUs are still useful for short periods of time. For example, figure 1.2 is an example of a short rocket launch that was tracked via a MIMSY. Advances in camera sensors could also allow vision to be a legitimate way to navigate microrockets. For example, the AMS NanEye[2] camera is a 1 mm x 1 mm footprint 62 kilopixel camera, which fits the payload budget limitations of a microrocket system. Both of these sensing modalities could be interfaced with the SCμM SoC, providing a feasible computation platform for a microrocket.

[2]`https://ams.com/naneye#tab/features`

Actuation

Modes of actuation are essential for controlled microrockets. A common method by which atmospheric rockets are actuated is via aerodynamic control surfaces. These usually manifest themselves as airfoils that can be rotated with respect to the direction of travel. As these surface are rotated, they generate lift forces which can be used to alter the trajectory of a rocket. Missiles commonly use actuated canards, or airfoils in front of the main fins of the missile, to control their flight. Currently, there is a dearth of miniaturized actuators for microrockets.

Fortunately, developments in micro-electromechanical systems (MEMS) technologies have made microscale actuators and miniature control surfaces possible. For example, Ho et al. were able to demonstrate aerodynamic control of a meter-scale delta wing by using an array of MEMS actuators to manipulate the flow separation at the leading edge of the wing [43]. Murthy and Krishna also developed pneumatically-actuated microballoon actuators for similar flow separation use [44]. Lyshevski further demonstrated the use of MEMS actuator arrays for control surfaces by designing a smart MEMS array that could modify the surface geometry of a wing into a control surface [45]. Additionally, Low et al. demonstrated the trajectory control of a hypersonic projectile by using pressure-actuated MEMS diaphragms [46]. Some MEMS control surfaces used arrays of MEMS actuators to manipulate airflow over a centimeter-scale to meter-scale delta wing [47, 48, 49]. These MEMS arrays aren't suitable for PAVs in their current form because their array size is too large. In [17], Wood et al. developed miniature piezoelectric actuators that were eventually used to actuate the ailerons on a 2 gram microglider [18]. This control surface is the most similar to the type of control surface that would be useful for our microrockets. Millimeter-scale control surfaces that are suitable for PAVs will resemble control surfaces like these.

Miniature control surfaces would also be an enabling technology for pico air vehicles, which are unmanned aerial vehicles (UAVs) that are smaller than 5 cm in any dimension and lighter than 500 mg [50]. By combining recent advances in electrohydrodynamic (EHD) thrust with MEMS aerodynamic control surfaces, we hope to develop centimeter-scale EHD-propelled pico air vehicles in parallel with millimeter-scale rockets [51]. This work will focus on developing millimeter-scale actuators that are suitable for microrockets and PAVs.

1.2 Inchworm motors as an Actuation Modality

While significant progress has been made on miniaturizing computation, communication, sensing, and power systems, further work is needed on control and actuation modalities. In this work, we propose a MEMS control surface that utilizes electrostatic inchworm motors to rotate a thin silicon fin. Electrostatic inchworm motors are one class of linear MEMS actuators that have a favorable tradeoff between force density and displacement [21], [22]. These motors rely on two sets of electrostatic gap closing actuators (GCA) that actuate out of phase of each other. Each phase of actuation produces a few microns of displacement,

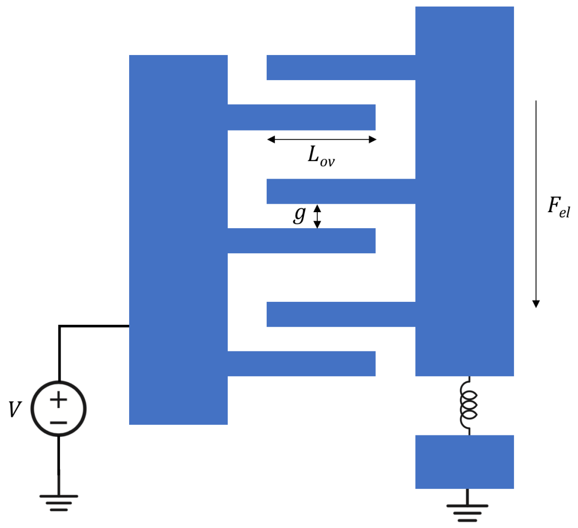


Figure 1.3: Electrostatic gap closing actuator diagram.

but as these actuators repeatedly operate, these small steps add up to a large displacement. Inchworm motors can be fabricated using lithographic techniques widely used in MEMS fabrication processes. While these motors have been used to actuate microrobots [6] [9], they have yet to be applied to microactuated aerodynamic control surfaces. Additionally, planar silicon-on-insulator (SOI) pin joints [52] can be used to design mechanisms that translate linear inchworm actuation to more complicated motions like rotation [6], which is required for the aerodynamic control surface. We direct the interested reader to [21, 53], which provide in depth description and analysis of the inchworm motors used in this MEMS control surface work.

The force generated by and electrostatic GCA is described by Equation 1.4, where ϵ_0 is the free space permittivity constant, N is the number of comb fingers, V is the operating voltage, and g is the gap width between electrodes. The electrostatic attraction between the two components of the actuator generates the force. Figure 1.3 illustrates the geometry and operation of a GCA. In reality, the second, larger gap does have a small impact on the force generated by the actuator, but since this effect scales with the square of g and $g_{large} >> g_{small}$, this effect is negligible. The GCAs commonly used in inchworm motors require 25V-100V to operate.

$$F_{el} = \frac{1}{2}\epsilon_0 \frac{NV^2}{g^2} \tag{1.4}$$

The GCA is the primary functional component of an inchworm motor. An inchworm motor unit cell contains four GCAs, where there are two pairs of motors that are driven by the same control signal and thus are synchronized. Figure 1.4 illustrates the configuration of these GCAs within an inchworm motor and the inchworm motor's operation. This particular inchworm motor design has 45 ° angled arms that will transfer the inchworm force, which is perpendicular to the shuttle, to longitudinal force along the shuttle. The actual force that is transmitted to the inchworm shuttle is described by Equation 1.5, where F_{load} is the force applied to the inchworm shuttle, F_y is the force the GCAs produce, θ is the angle of the motor arm in relation to the shuttle, Δy is the the distance the GCA travels, and L_{arm} is the length of the angled arm [21].

$$F_{load} = \frac{F_y}{tan(\theta)} - k\frac{\Delta y tan(\theta)}{L_{arm}^2 sin^2(\theta)} \tag{1.5}$$

1.3 Power Supply

Many of the preceding system components require a supply of electrical power. For example, SCμM requires up to roughly 1 mA of current at 1.8V, the ICM-20948 requires up to 3 mA of current at 1.8 V, and the inchworm motors will require 10 - 100 μA at up to 100 V to operate effectively. While miniature batteries can supply the lower voltage components, the high voltage required by the inchworm motors can pose a challenge for miniaturized electronics platforms to supply. Recent advances in high voltage solar power supplies provide a potential solution to this problem. The Zappy chip [54] is a solar cell power system that can provide high voltages of 100 V by combining hundreds of individual cells in series. This system also includes lower voltage power rails that can supply integrated circuits like SCμM and an IMU. Moreover, the Zappy chip has been demonstrated to successfully power a SCμM and inchworm motor gripper system, which indicates the viability of powering a SCμM-based microrocket system.

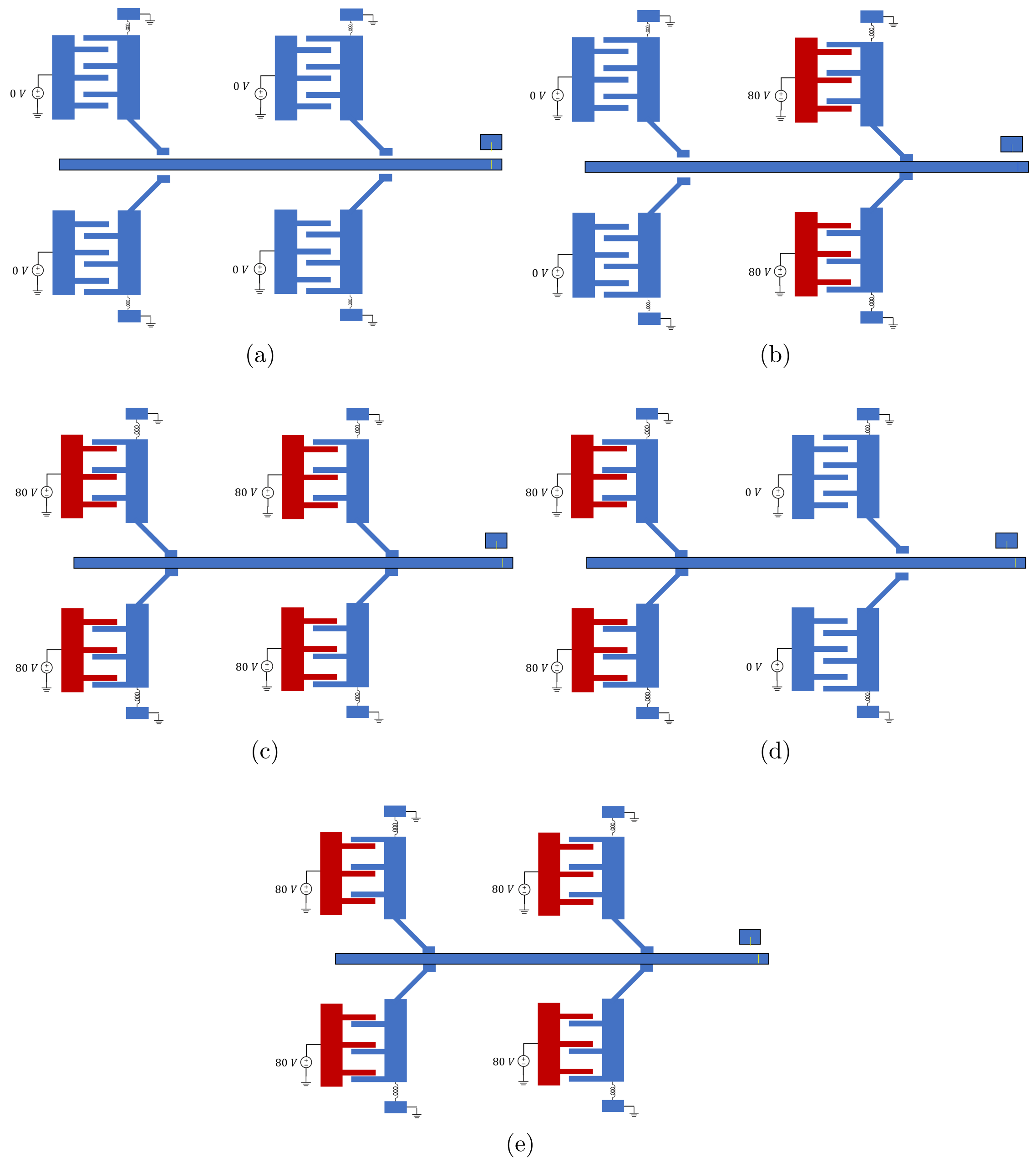


Figure 1.4: Diagram of inchworm motor operation. (a.) Inchworm motor at rest with both sets of GCAs grounded. (b.) One pair of GCAs actuates, contacting and pushing forward the inchworm shuttle. (c.) The second pair of GCAs actuate, which will allow the first pair to release without the shuttle losing ground. (d.) The first pair of GCAs release which allows the second pair of motors to push the shuttle forward. (e.) The first pair of GCAs re-engage, and the processes (b-d) repeat, pushing the shuttle forward.

Chapter 2

Inchworm Motor Control Surfaces[1]

2.1 Analysis and Design of MEMS Control Surface

In order to design viable MEMS control surfaces, we first determined the amount of force required to maneuver a millimeter-scale rocket with control surfaces. We then used this force estimate to determine the required fin area and actuator torque for the control surface, which largely determined the design of the actuator and fin.

Simulated Control Performance

Most aircraft use control surfaces such as ailerons, elevons, and rudders to control their trajectories. Thin airfoil theory, can be used to estimate the lift force generated by an airfoil that is rotated by an angle of attack with respect to wind-relative velocity [55]. Equation 2.1 describes the lift force generated by a thin airfoil with a variable angle of attack, α. This approximation holds for $|\alpha| < 15°$.

$$F_{lift} = \frac{1}{2}\rho V^2 2\pi\alpha A_{ref} \tag{2.1}$$

$\frac{1}{2}\rho V^2$ describes the dynamic pressure of the airflow surrounding the fin, where ρ is the density of the surrounding air, and V is the speed of the airflow. A_{ref} is the area of the fin in question. As Equation 2.1 shows, altering the angle of attack of the control surface allows us to linearly control the aerodynamic lift force generated by the surface. In order to remain effective, the control surface must be able generate enough torque to resist this lift force and maintain its angle of attack. Consequently, we must design our actuators to meet this torque requirement.

The aerodynamic lift force generated by the control surface applies a torque to the rocket. This provides us with a method of controlling the pitch, roll, and yaw of the rocket. For the pitch and yaw axes, this moment is described by Equation 2.2. In this equation, d_{cp} is the

[1]This chapter is adapted from [37, 38]

distance between the control surface and the center of gravity of the rocket. For the roll axis, the torque is described by Equation 2.3, where r is the radius of the rocket and r_{cp} is the distance from the base of the fin to the center of pressure of the fin.

$$M_{y,z} = F_{lift} d_{cp} \tag{2.2}$$

$$M_x = F_{lift}(r + r_{cp}) \tag{2.3}$$

This moment induces a rolling acceleration that alters the rocket's trajectory. The magnitude of this acceleration is determined by the inertia tensor of the rocket, which can be approximated by the inertia tensor of a cylinder (Equation 2.4).

$$I = \begin{bmatrix} \frac{1}{12}mh^2 + \frac{1}{4}mr^2 & 0 & 0 \\ 0 & \frac{1}{12}mh^2 + \frac{1}{4}mr^2 & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix} \tag{2.4}$$

In this expression, m is the mass of the cylinder, and h is the height of the cylinder. Assuming that m scales with S^3, where S represents a single dimension, we find that I scales with S^5. Compared to the control surface force, which scales with S^2, I scales very favorably when size is decreased, which allows us to scale down these rockets while increasing maneuverability. In order to illustrate the MEMS control surfaces' potential utility, we simulated the trajectory of a 3 mm x 6 cm rocket with 8 mN thrust and controlled by MEMS control surfaces with 2 mm x 4 mm airfoils. Our simulations show that the rocket is able to perform 18m radius turns at 20 m/s (which is 2 g's of acceleration) when the control surfaces were generating 5 mN of lift per surface (Figure 2.1). These simulations show that 5 mN aerodynamic forces are sufficient for controlling the trajectory of a millimeter-scale rocket. This analysis shows us that we can obtain useful maneuverability from just millinewton-scale aerodynamic forces. Fortunately, electrostatic inchworms can easily meet these force requirements, as they have force densities of on the order of $1 \frac{\text{mN}}{\text{mm}^2}$ [21] and have been shown to produce up to $2 \frac{\text{mN}}{\text{mm}^2}$ [56].

Actuator Design

In our MEMS control surface, electrostatic inchworm motors actuate a rotary slot mechanism that includes a large slotted arm. A 2x4 mm^2 SOI fin is inserted post-fabrication into this slotted arm. Figure 2.2 shows the layout of the inchworm motors and rotary slot mechanism, which is fabricated in a SOI process. Two inchworm motor arrays are used to to achieve bidirectional rotation; as one motor actuates the other relaxes and allows the shuttle to pass freely through it. The rotary slot mechanism uses silicon pin joints [52] to translate the linear actuation of the inchworm motors to a rotation in the fin slot arm. Additionally, the rotary slot mechanism provides mechanical advantage for the system, so geometric changes in the layout can be made to accommodate different design specifications. The mechanical advantage for our mechanism is $\frac{d_{arm}}{d_p} = 1.36$; d_{arm} is the length of the arm linking the inchworm shuttle to the slotted arm, and d_p is the distance between the slot's fulcrum and the center of pressure

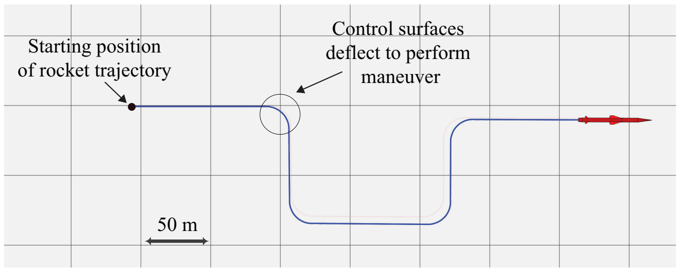


Figure 2.1: Simulated trajectory of a millimeter-scale rocket powered by a 5 mN thrust and controlled by MEMS control surfaces. The simulated rocket travelled almost 400 m at a speed of 20 m/s and demonstrated a turning radius of approximately 20 m.

of the inserted fin. Note that this value can be zero, if the center of pressure of the fin is directly above the pin joint. If d_p is positive, the fin will have an equilibrium state at $\alpha = 0$ if the actuators are not powered.

As this fin is 40 μm thick and the maximum angle of attack of the actuator is less than 15°, Equation 2.1 accurately models this control surface. Using calculations from [21], the inchworm motors should have a theoretical force density of 2.11 $\frac{\text{mN}}{\text{mm}^2}$ while running at 110V, which yields 2 mN maximum output force for each 1 mm^2 inchworm motor. We designed this control surface to have a maximum angle of attack of approximately ±10° and a maximum output torque of 6 μNm. This output torque corresponds to an output force of 5 mN at the fin's center of pressure, which is located at $\frac{1}{4}$ the length fin's chord length. Consequently, this fin should be able to handle a 5 mN aerodynamic lift force while maintaining its angle of attack, which is necessary for the control surface to remain effective.

We fabricated the MEMS control surfaces in a silicon-on-insulator (SOI) wafer. The SOI fabrication process is illustrated in Fig. 2.3. Additionally, we evaporation-deposited a 200 nm layer of aluminum in order to metalize the pads for wirebonding. After fabrication, we assembled the fin structure by inserting the fin into the fin slot and securing it with silver epoxy. We then mounted the structure on a flex-printed circuit board (PCB) and wirebonded the PCB pads to the metalized control surface pads. Figure 2.8 shows the fully assembled control surface system. External wires were soldered to the PCB pads and used as power lines for the inchworm motors.

2.2 Preliminary MEMS Control Surface Design V1

The MEMS control surface originally was designed to have a much thinner fin slot, which also required inserting the fin into the fin slot vertically. Not only was the slot too thin to survive fin insertion, but the fin insertion process itself was difficult to accurately place the

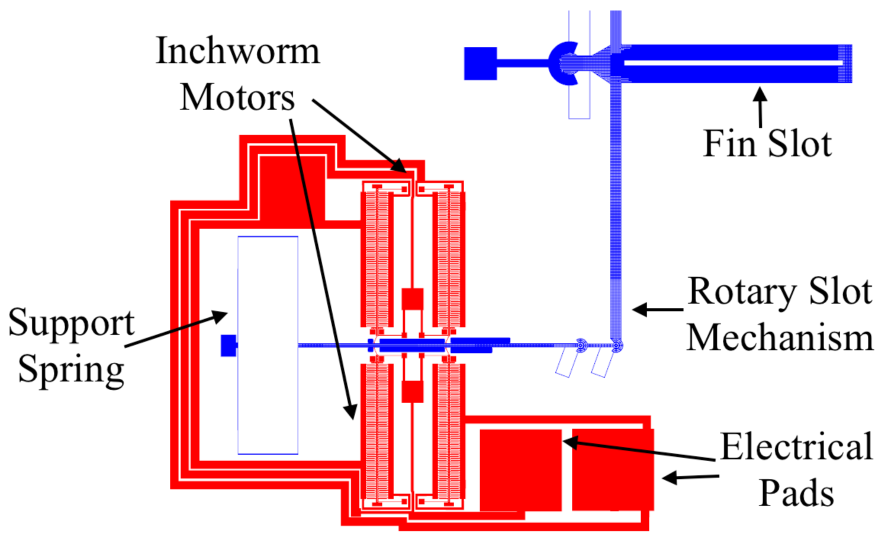


Figure 2.2: Layout view of the bottom half of the MEMS control surface chiplet showing inchworm motors in red and the rotary slot mechanism in blue. The top inchworm motor is symmetric with the bottom one. The control surface chiplet is 7 x 9 mm^2.

fin in the 45 μm slot. This device also did not have sufficient inchworm shuttle guides, which led to inchworm pawl jamming (Figure 2.5). As the the fin slot deflected, force was applied by the pin joints perpendicular to the inchworm shuttle, causing lateral deflection. Without shuttle guides, this deflection was great enough for the inactive inchworm motor's shuttle to come into contact with the inchworm pawl, locking the entire assembly and preventing the active inchworm motor from further deflecting the fin slot.

2.3 MEMS Control Surface V2: Improving Robustness

The second iteration of the design prevented pawl jamming by adding additional shuttle guides to the inchworm motors (Figure 2.7). These guides constrained the transverse movement of the inchworm shuttle, preventing its teeth from catching on the unactuated inchworm pawl. The solid beams on either side of the fin slot were also increased in thickness from 50 μm to 140 μm, which increased its robustness. These modifications were enough to get the device working in a rudimentary way. Assembly was difficult and low yield. The full assembly can be seen in Figure 2.8. The electrical contact pads were increased in size and metallized to facilitate wirebonding.

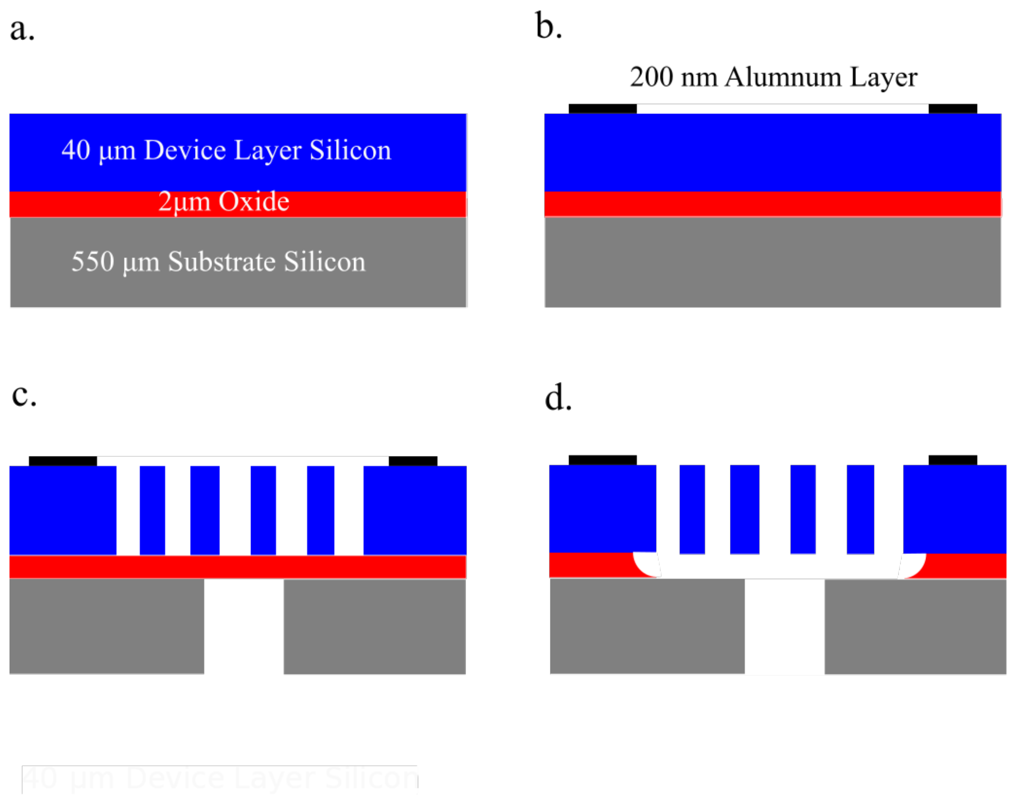
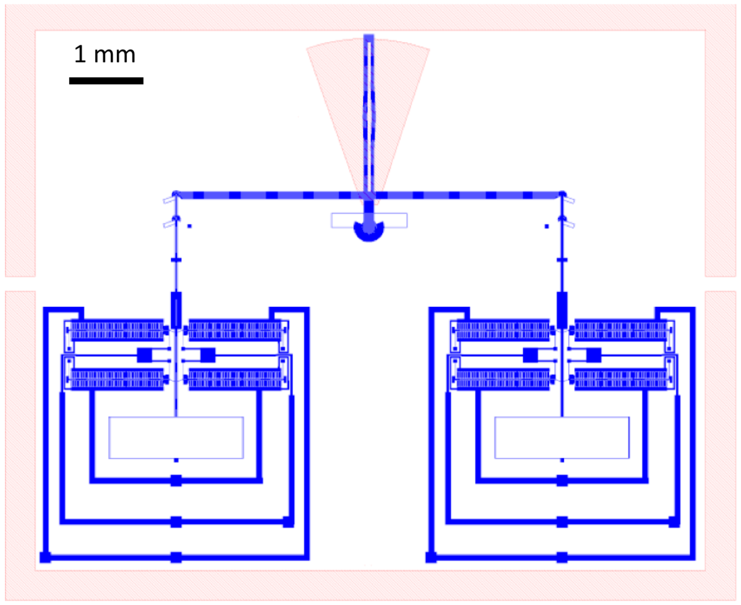


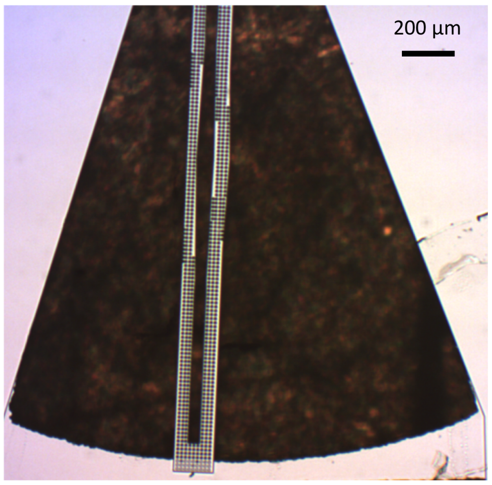
Figure 2.3: SOI fabrication process. (a.) SOI wafer before processing. (b.) SOI wafer after evaporation deposition of 200 nm aluminum pads. (c.) Deep reactive ion etch (DRIE) patterns inchworm motors and pin joints in device layer silicon and a backside DRIE etches trenches in the silicon substrate (c.) Hydroflouric acid vapor etch releases device layer structures from the oxide layer.

2.4 Force Characterization Experimental Setup

We performed both functionality and force characterization experiments for the control surfaces. Displacement angles of the fin slot were determined visually by capturing images with a microscope camera and measuring the angular displacement using digital image editing software. We designed force gauge test structures attached to the fin slot to aid the characterization of the force output (see Figure 2.10). This test structure is attached to the same position on the fin slot as where the center of pressure of the inserted fin would be located, so the force applied by the force gauge should map equivalently to an aerodynamic lift force generated by the fin. In order to measure the output force of the fully assembled control surface, we placed a mounted needle on a scale and rested the fin on the needle. We then actuated the fin and measured the change in weight on the scale. We used an Arduino Uno with high voltage MOSFETs to generate the necessary inchworm motor control signals from a 0-120V DC power supply.



(a)



(b)

Figure 2.4: (a.) Layout of the first iteration of the MEMS control surface design. This iteration was not functional due to a weak fin slot and inchworm pawl jamming. (b.) The fin slot was too weak to survive assembly and the insertion of the fin into the slot.

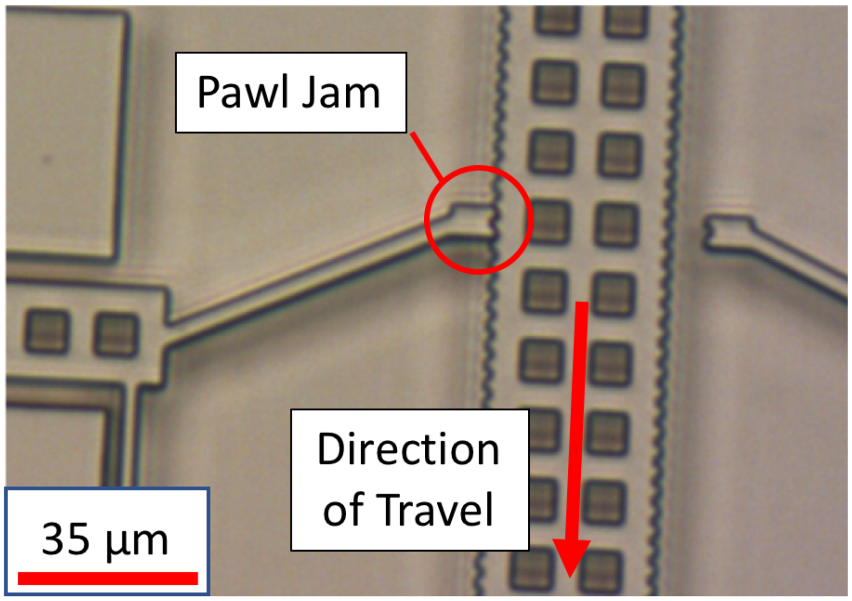


Figure 2.5: Image of inchworm motor shuttle jamming on pawls. This set of inchworm motors is unpowered and should be allowing the shuttle to move freely past its pawls.

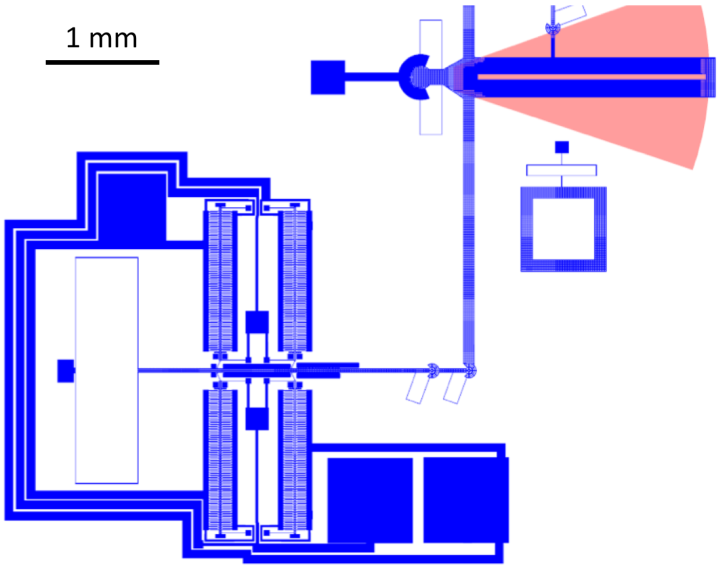


Figure 2.6: Layout of the second version of the MEMS control surface.

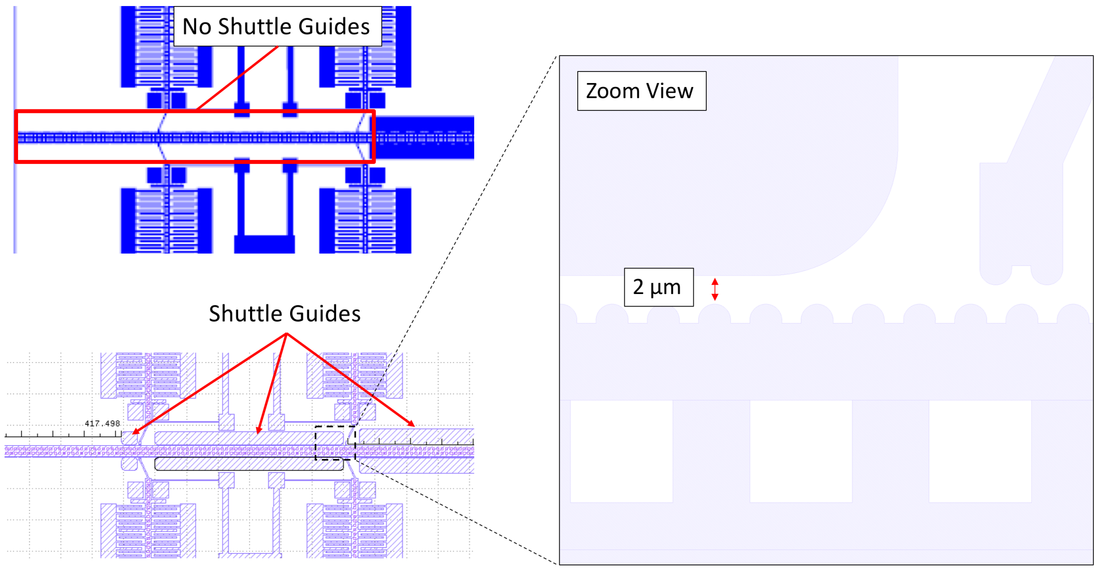


Figure 2.7: Comparison of insufficient shuttle guides from Version 1 (Top) and sufficient shuttle guides from Version 2 (Bottom). The design from Version 1 did not have enough guides near the support spring, so the shuttle had enough freedom to move laterally as well as longitudinally. The second design version thickened the fin slot and added shuttle guides to prevent the inchworm shuttle from catching on unpowered inchworm motor pawls.

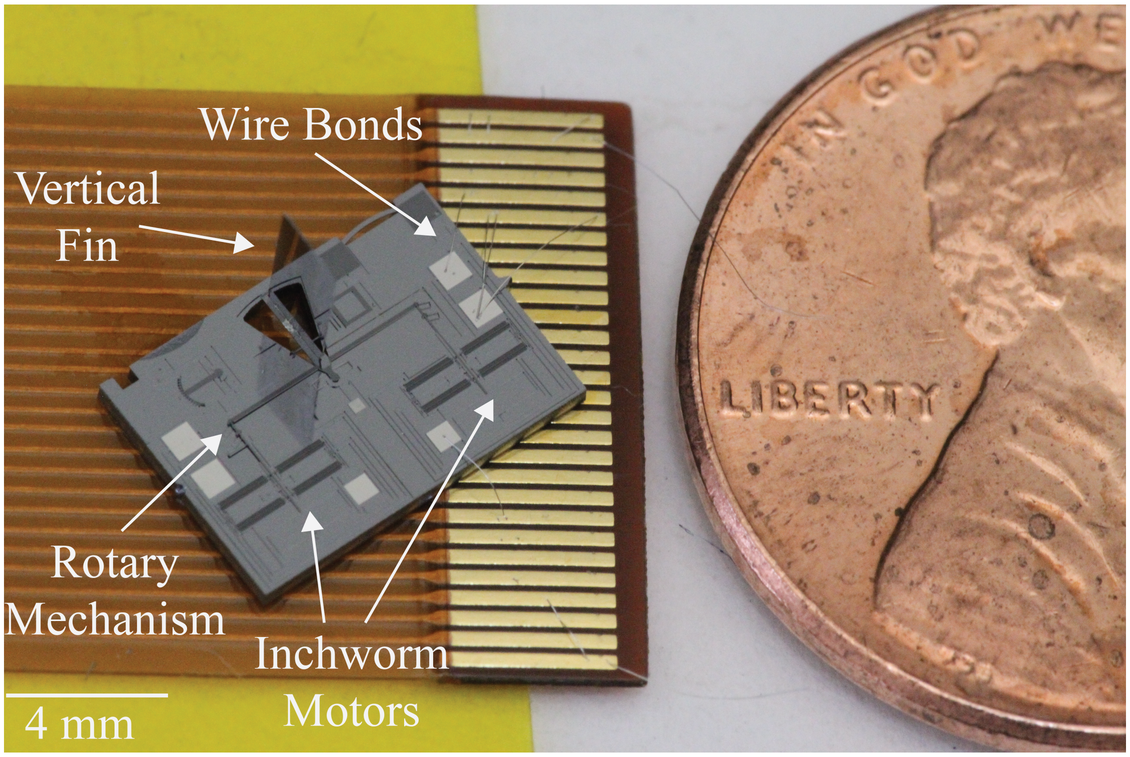


Figure 2.8: Fully assembled and wirebonded fin. The total size of this assembly is 7 mm x 9 mm x 3 mm.

2.5 Experimental Results

We were able to achieve maximum deflections of 10° in the fin slot while operating an unassembled control surface at a probe station. Figure 2.9 illustrates this actuation. The maximum speed these inchworm motors could be run before failing to actuate was 3.5 kHz, which corresponds to a rotation rate of 100 degrees/second. This limit on drive frequency is most likely due to the RC time constant of the motor capacitance and the pull up resistors in the switching circuit. We could likely increase the maximum motor frequency by using more sophisticated switching circuits that would reduce the RC constant.

Additionally, we estimated the force output of actuator mechanism using our force gauge test structure. Force outputs were estimated by first measuring the angular displacement of the fin slot. We then derived the spring constant of the force gauge from its layout dimensions and used it to convert the displacement of the fin slot into a force. Results of this experiment are shown in Figure 2.10. The maximum estimated output force was 1.5 mN, which was reached at 80V. At 80V, the actuator reached its maximum deflection point, which saturated the measurement from the force gauge. Interestingly, the estimated actuator force was not quadratically related to voltage, which is how theoretical electrostatic actuators normally behave. This phenomenon could be explained by an increased electrostatic attraction between

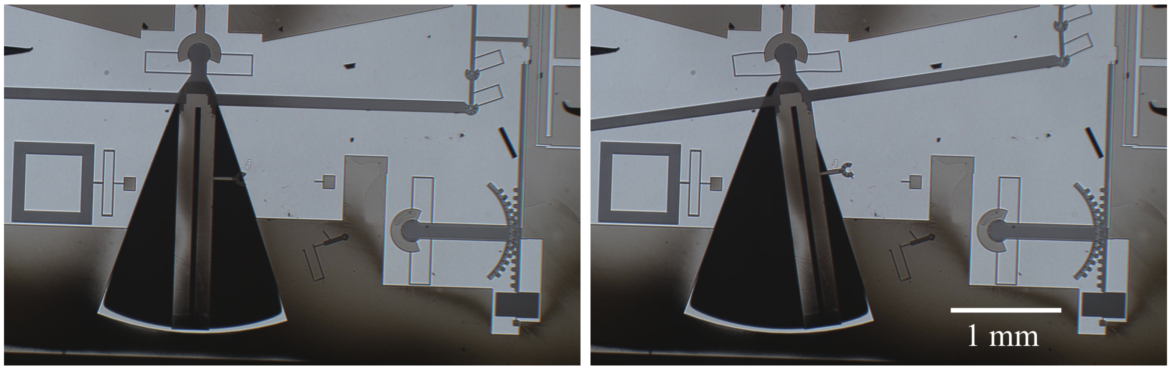


Figure 2.9: Top: Rotary slot mechanism at no deflection. Bottom: Mechanism at maximum deflection of 10°. This motor was being driven at 90V and a frequency of 100Hz.

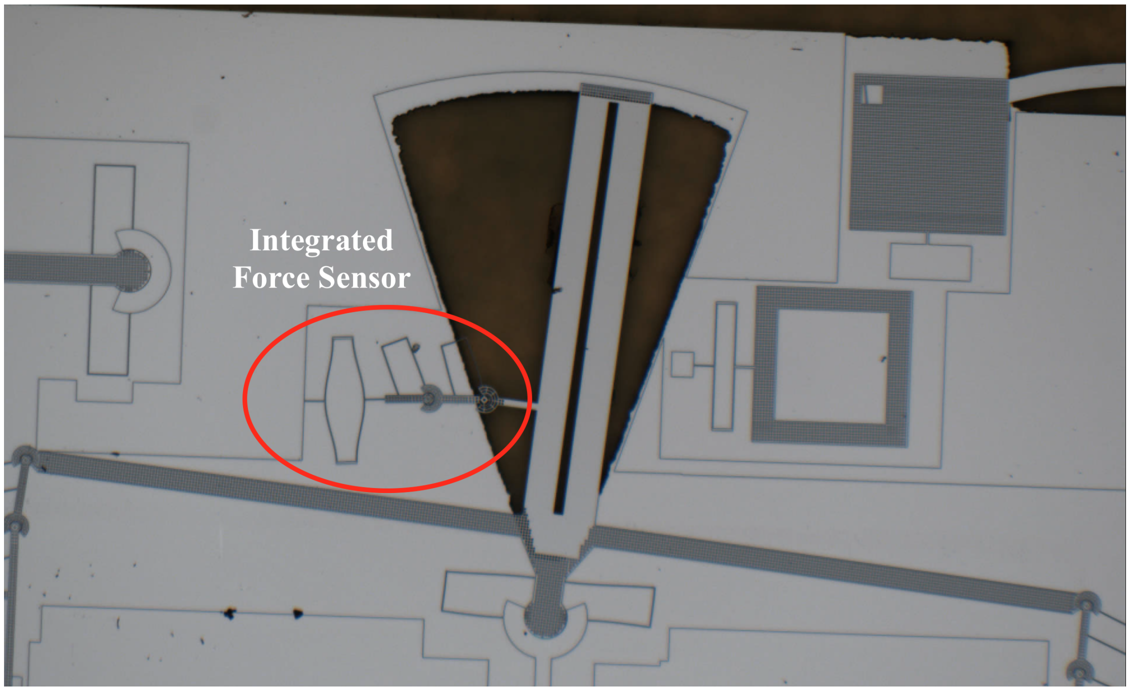
the substrate and a released rotary slot mechanism that are at different voltages or it could be due to unexpected friction forces. While these structures are both designed to be at the same ground potential, they could easily develop a voltage differential if small conductive debris, such as silicon shards, create a weak electrical connection between an electrical drive trace and another structure.

We also demonstrated the successful actuation of the fully-assembled control surface (see Figure 2.11). The assembly allowed us to drive the control surface outside of a probe station, which is necessary for independent operation on an miniature aerial vehicle like a millimeter-scale rocket.

Finally, we characterized the force output of the fully assembled control surface (see Figure 2.13. The maximum measured force was 1.05 mN at a distance of 1500 μm from the fulcrum, which corresponds to a maximum torque of 1.575 μNm. The measured output force of the fully-assembled control surface is similar to the measured output force measured by the integrated silicon force gauge when differences in mechanical advantage are taken into account. Additionally, the output force of the fully-assembled control surface is not quadratically dependent on input voltage. At a maximum output torque of 1.575 μNm, the assembled control surface performs at 26.3% of its expected performance. This could be attributed to stiction, friction, and parasitics in the structure. We can address this performance discrepancy in future designs by increasing the mechanical advantage of the rotary slot mechanism or increasing the inchworm motor area to increase torque.

2.6 MEMS Control Surface V3: Designing for Assembly

At this point of development, the MEMS control surface in [37] had several problems, including lack of robustness to out of plane motion, difficulty of assembly, and low assembly yield. This fin also had not been tested in a dynamic airfoil, so it was still uncertain if it could generate aerodynamic lift forces, which would be required to control a miniature rocket.



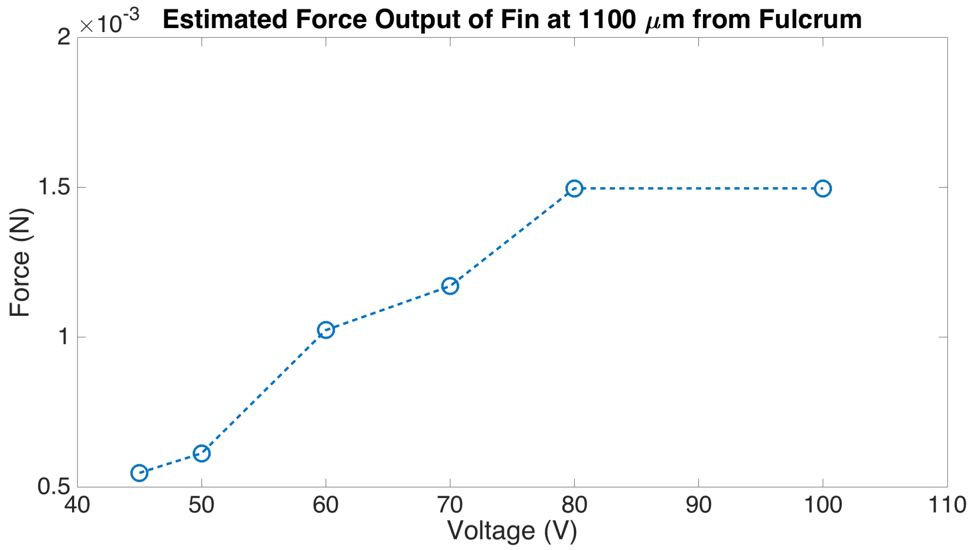


Figure 2.10: Top: Integrated force gauge experiment used for estimating output force of control surface. Bottom: Estimated force vs. voltage plot for the MEMS control surface. We optically measured the displacement of the force gauge and calculated the applied force by using the simulated spring constant of the structure.

While version 2 improvements allowed the system to operate successfully, the assembly process was still extremely difficult to perform effectively. The goal of the third revision was to improve the ease-of-assembly of the control surface. The primary focus was on the

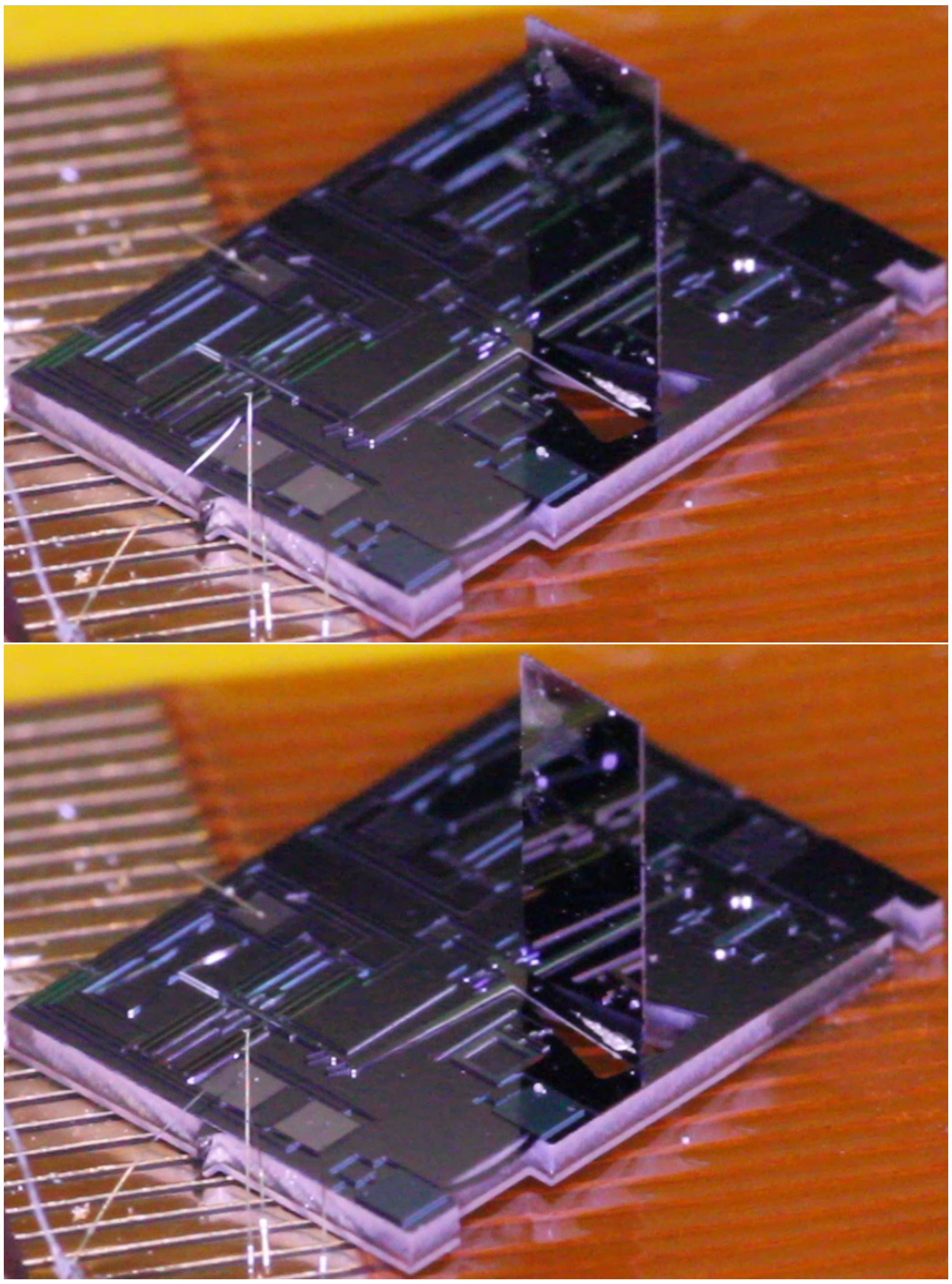


Figure 2.11: Demonstration of fully assembled fin actuating. Top: Fin at rest. Bottom: Fin at a deflection of 5°.

fin insertion process, which originally was performed in a top down manner. This assembly method required precise angular alignment and careful hands to avoid fracturing the fin slot arm. Version 3 improved this process by using a split slot arm that facilitated inserting

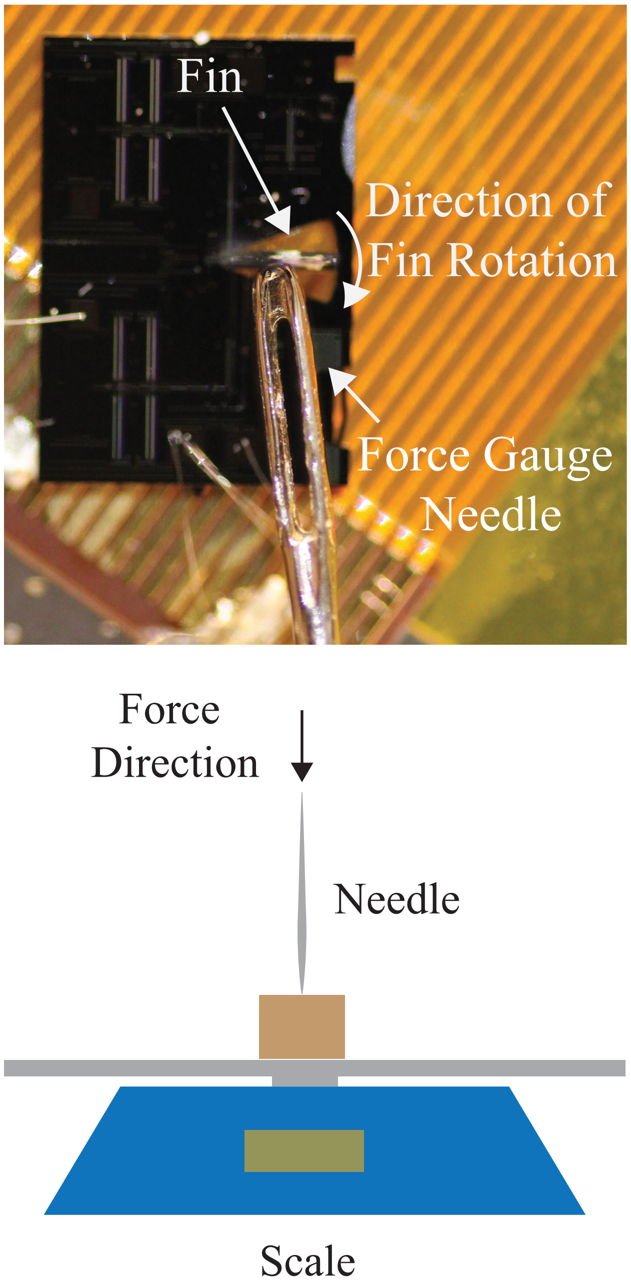


Figure 2.12: Left: Force gauge tip placed underneath the fin. This needle was oriented vertically and the fin would rotate down into it. As the fin actuated downward, it pressed down on the needle, and the resulting force was measured by a scale below. Right: Illustration of needle-scale force gauge.

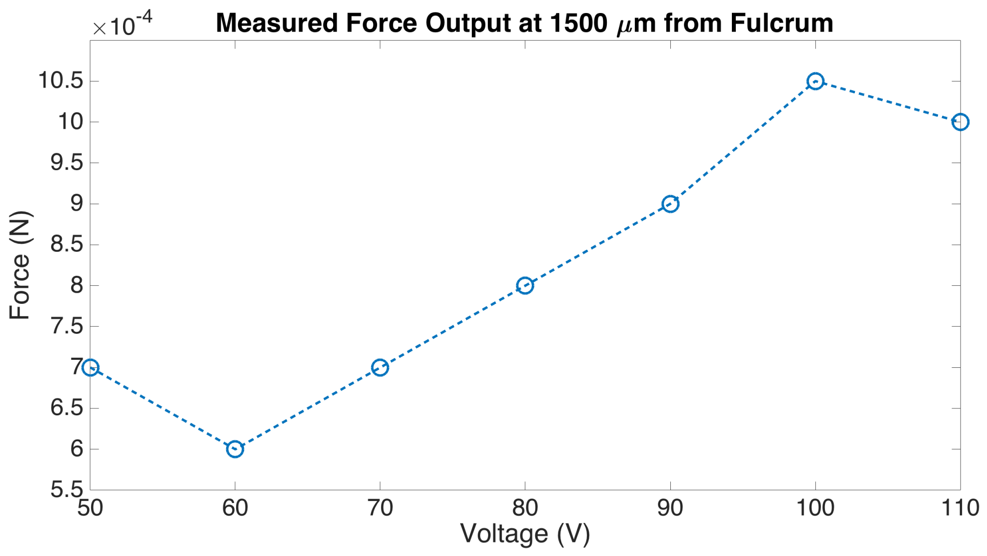
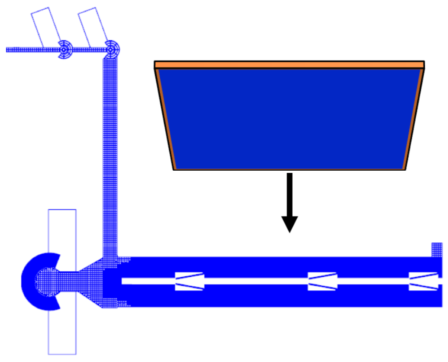


Figure 2.13: Measured force vs. voltage plot of the fully-assembled control surface. The inchworm motors were operating at a frequency of 500 Hz when this data was taken.

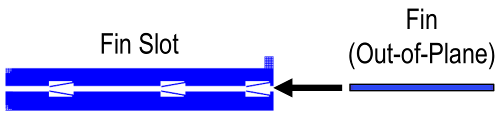
the fin from the side instead of from above (see Fig. 2.14). This design was also somewhat self-aligning, which further improved ease-of-manufacture for the fin insertion process. A leftover silicon leg (a silicon beam) from [6] was used to prevent the fin arm from moving out-of-plane and popping out of the socket. This part was silver epoxied to a portion of the SOI layer that was far from any moving parts. While this was intermittently successful, the silver epoxied caused mobility problems for the mobile parts. As the silver epoxied was heated for curing, a sticky substance wicked out of the epoxy and would spread to mobile parts on the device. Parts affected by this were stuck to the substrate and were immobilized. This step of the assembly process was the most difficult and had the lowest yield. The wirebond pads were moved to make it easier to bond from a mounting PCB to the chip.

This control surface could be improved by an integrated force measurement system. Integrated force sensing eliminates the need for complicated and cumbersome external force measurement systems, and it enables force feedback applications. Integrated force measurements can be used to improve microbotic control systems. For example, B. Yang et al. used simulated motor force outputs along with machine learning and dynamic simulations of a microrobotic hexapod to design an optimized hexapod gait [57]. Integrated motor force sensors will be necessary in order to evaluate and tune control schemes like these when they are implemented on a physical microrobot.

Several piezoresistive and capacitive force sensors for microrobotic applications have been developed [58, 59, 60]. The systems in [59] and [60] both were fabricated in an SOI process compatible with an SOI inchworm motor system. Xu et al. integrated a capacitive force sensor with an electrostatically-actuated microgripper, which demonstrated the integration of force sensors with actuator systems in SOI processes [61].



(a)



(b)

Figure 2.14: Comparison of fin insertion methods. (a.) Top-down fin insertion, where the fin is inserted perpendicularly into the fin slot. (b.) Side insertion, where the fin is slid into the slot from the side and is somewhat self-aligned by a funnel structure in the device layer (not shown).

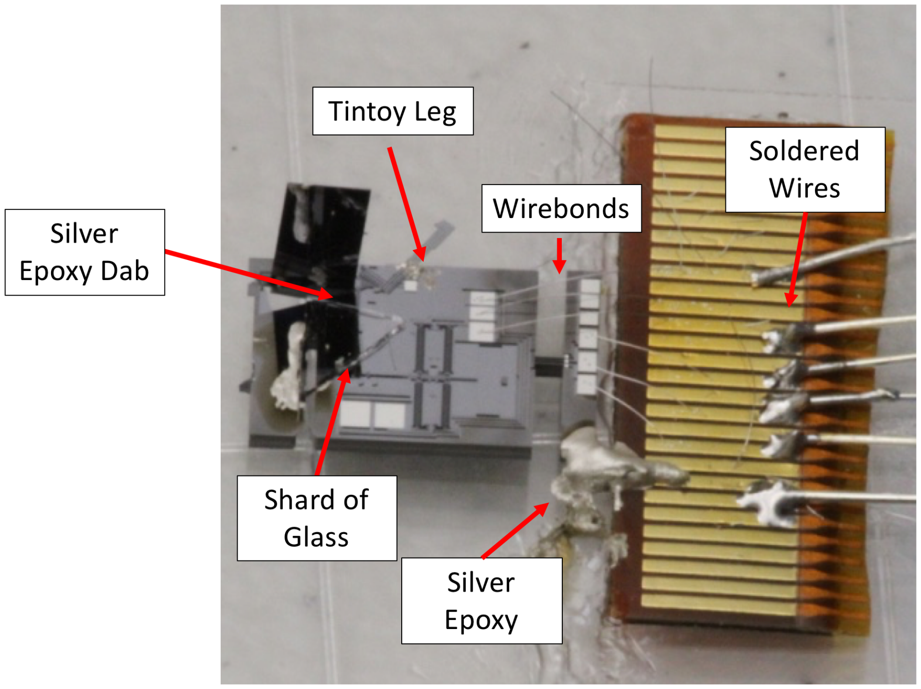
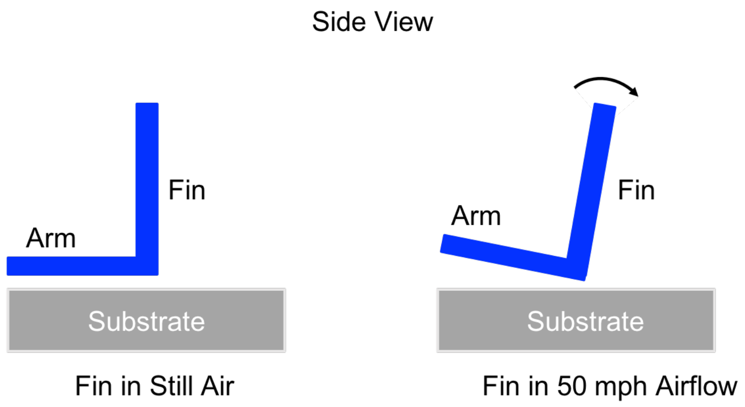
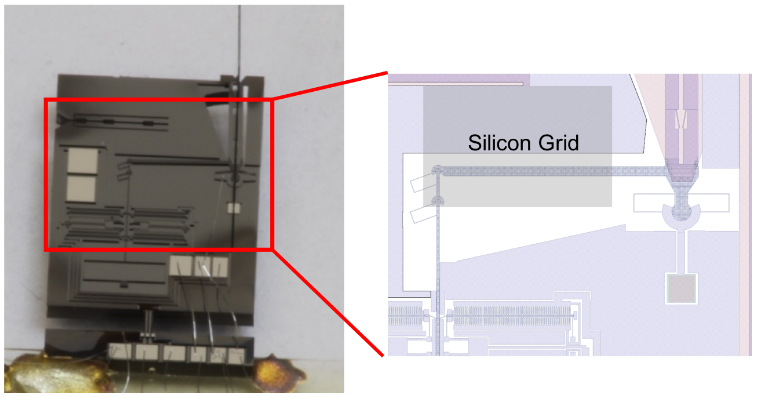


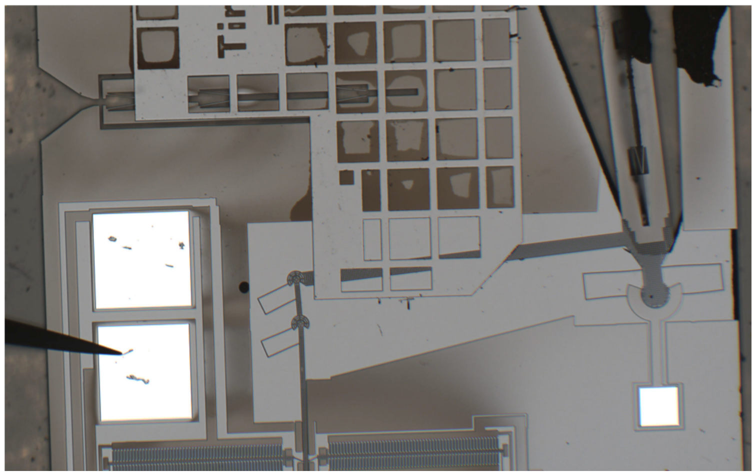
Figure 2.15: Fully assembled MEMS control surface



(a)



(b)



(c)

Figure 2.16: Plane stability solutions

This following sections are adapted from [38], which documents the results of improving manufacturability, increasing out-of-plane robustness, and performing wind tunnel tests. This paper describes the design of a MEMS control surface actuation system integrated into a piezoresistive force sensor using a simple SOI process. The SOI process included a 550 μm single-crystal silicon (SCS) substrate, a 2 μm oxide layer, a 40 μm SCS device layer, and a metallization layer. The process steps were a substrate deep reactive ion etch (DRIE), a device-layer DRIE, a metal deposition, and a sacrificial oxide etch.

2.7 Theory and Design

Aerodynamics and Airfoil Design

MEMS Mechanisms

The MEMS control surface consists of electrostatic inchworm motors and rotary joints to actuate a rotary airfoil slot (Figure 2.18), which are fabricated in the 40-um device layer.

In the previous design iteration of this device, airfoil assembly was a significant challenge and decreased the yield of the device assembly process. A self-aligning airfoil slot was designed to avoid this problem. Instead of inserting the fin into the slot from above, the fin is inserted form the side, while spring-loaded mechanisms align and secure the airfoil (Figure 2.19). A small amount of epoxy was used to permanently fasten the airfoil into the slot. This improvement decreased the failure rate of this assembly step to zero.

Integrated Force-Sensing Platform

The motor, mechanism, and airfoil are fabricated into a platform that has integrated SOI half-bridge strain gauges (Figure 2.20). When aerodynamic forces are applied to the airfoil, the strain sensor beam deflects and the strain gauge registers a change in voltage. The integrated force sensing platform allowed us to easily measure the aerodynamic forces generated by the airfoil and could enable force feedback control schemes in future PAVs.

Theoretically, the output voltage of the half-bridge strain gauge is $V_o = G\epsilon_{avg}V_x$ where G is the gauge factor, V_x is the differential bridge voltage, and ϵ_{avg} is the average strain throughout the length each of the gauge's piezoresistors. The following equation shows the relationship between force applied to the airfoil and the average strain in the piezoresistors of the strain gauge, where F_l is the load force, d is the distance between the neutral axis of the substrate beam and the piezoresistor, L_g is the length of the L_f is the distance between the point where the load force is applied by the airfoil and calibration weights and the end of the beam, E is the Young's modulus of silicon, and I is the moment of inertia of the beam:

$$\epsilon_{avg} = \frac{(F_l d(L_f + \frac{L_g}{2})}{EI} \tag{2.5}$$

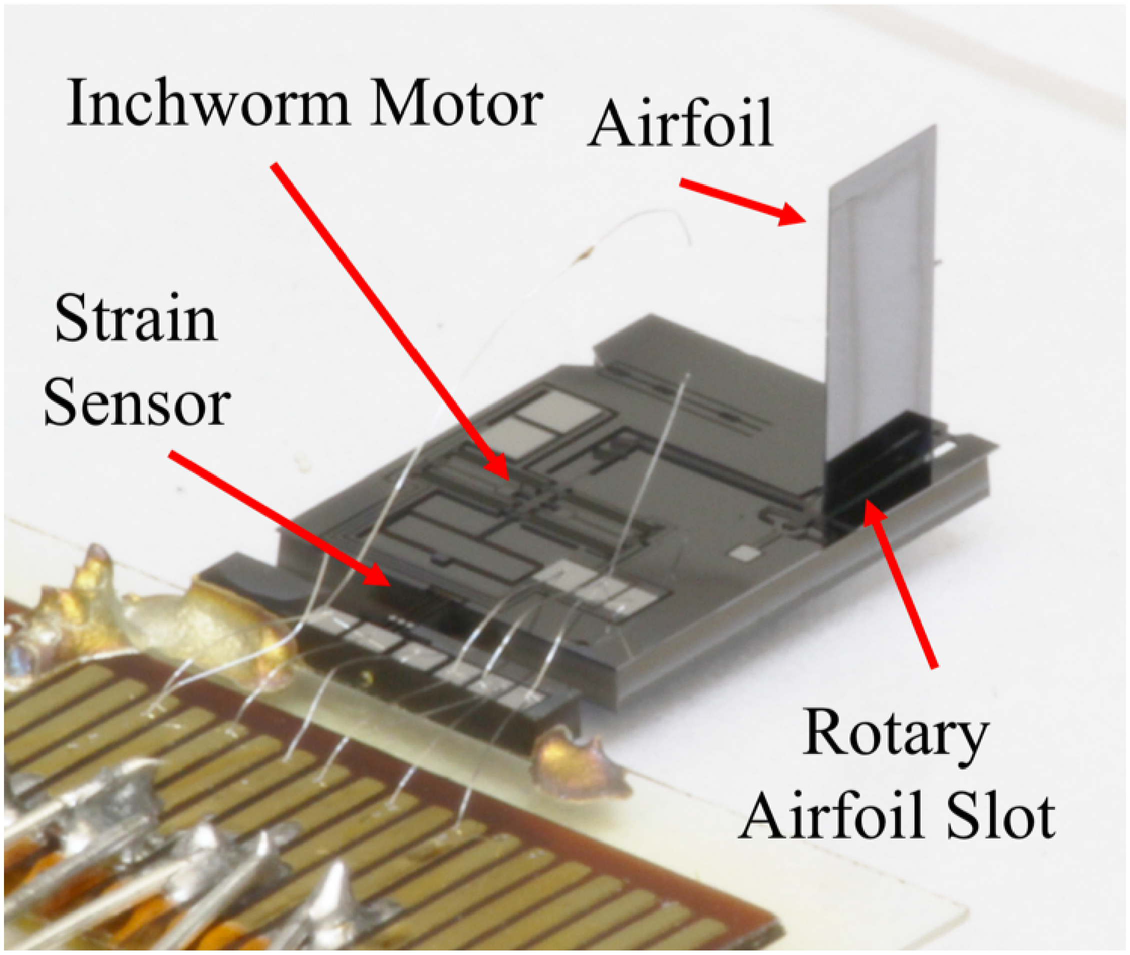


Figure 2.17: The MEMS control surface consists of an inchworm motor, a 2x4 mm^2 SOI device layer silicon airfoil, a rotary airfoil slot, and a piezoresistive half-bridge strain sensor. Six wire bonds are used to electrically connect the strain sensor and the motor. The silicon airfoil is inserted into the rotary airfoil slot and stands vertically out of plane from the rest of the device. The strain sensor measures generated lift force.

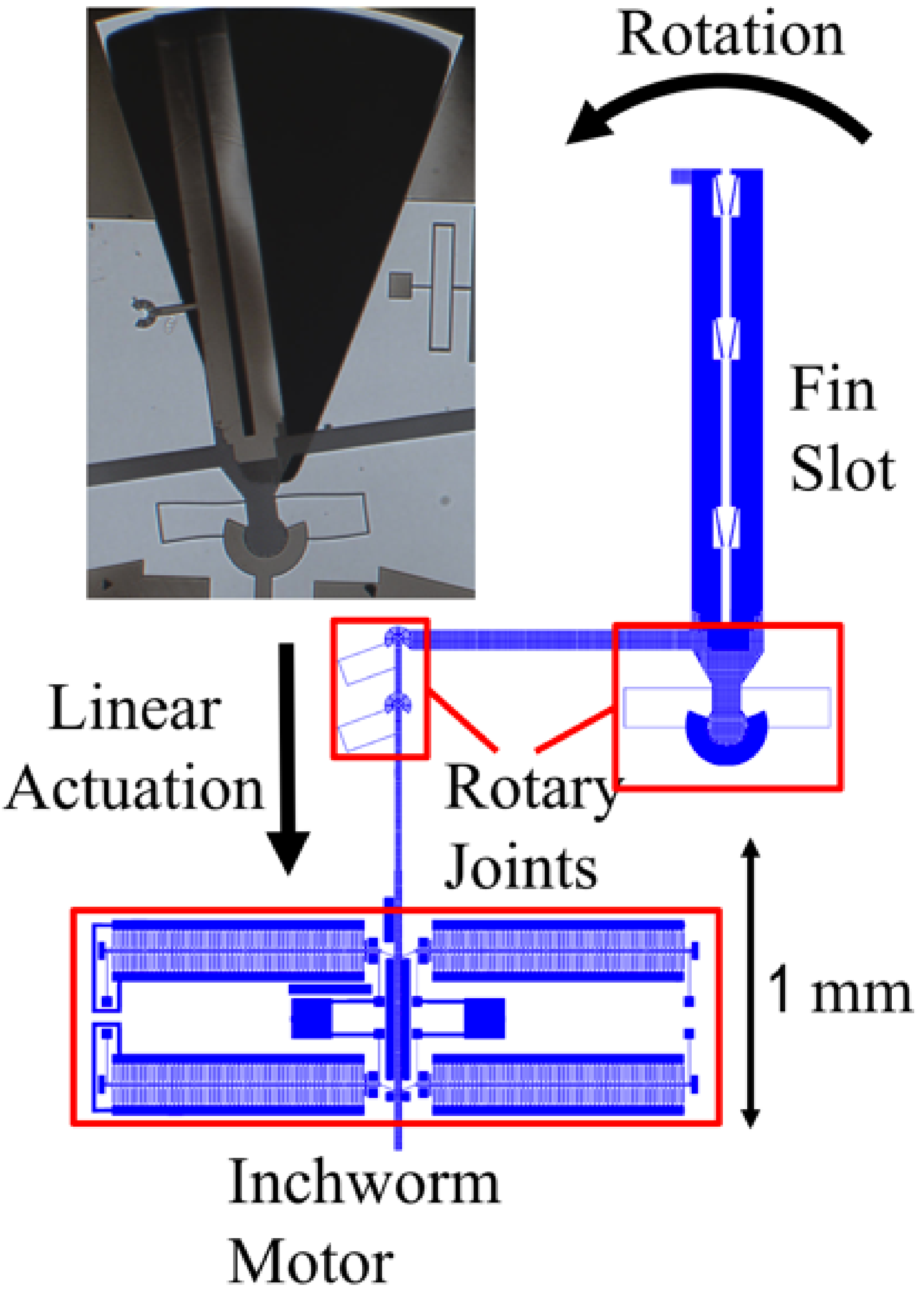


Figure 2.18: The inchworm motor linearly actuates the rotary joint mechanism which rotates the airfoil slot. The inset shows a microscopic photograph of the airfoil slot after rotation.

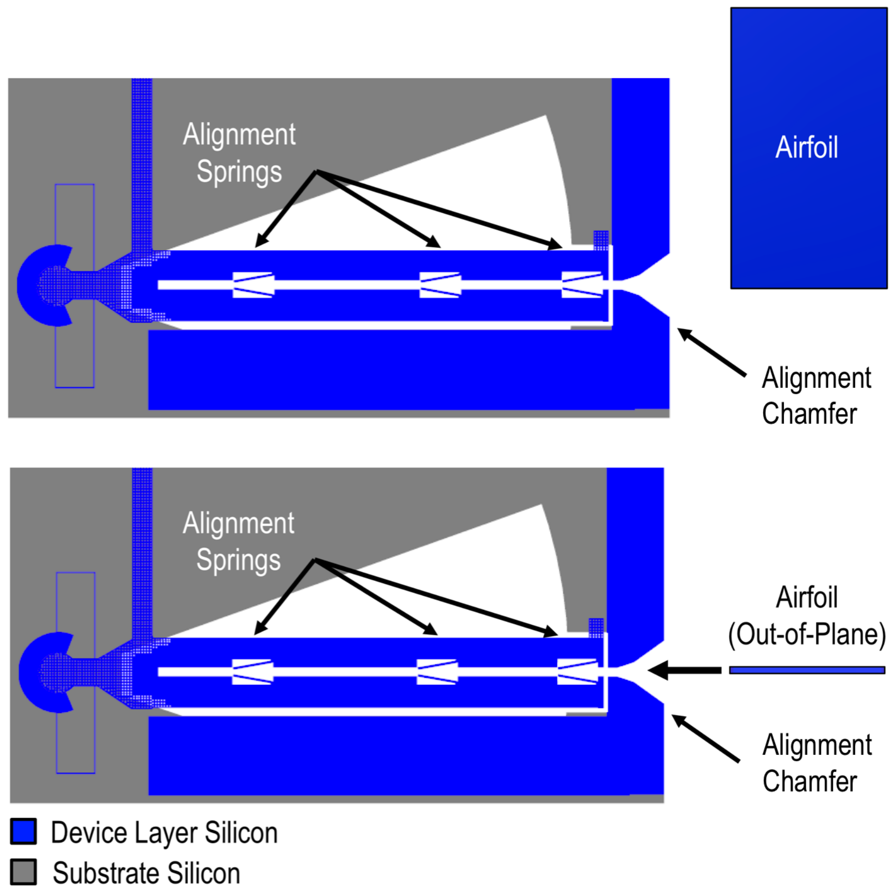


Figure 2.19: A patterned backside trench underneath the airfoil slot and a device-layer alignment funnel comprise the airfoil alignment mechanism. This mechanism guides the vertically-oriented airfoil into this slot as it is inserted. During airfoil insertion, cantilever springs align and secure the airfoil.

With $V_x = 1.63V$ this calculation predicts that the strain sensor will have a scale factor of 1.5 mV/mN. The strain sensor was calibrated by using known weights to apply force to the sensor and its output was measured through an 1000x amplifier. Vx was 1.64 V. Linear regression was performed on the calibration data, which yielded a linear fit with a slope of 1.96 V/mN and an R2 value of 0.986. This scale factor of 1.96 V/mN is 31 percent greater than the predicted 1.5 V/mN (adjusting for amplification) scale factor, and the average standard deviation of each measurement was 0.0175 mN.

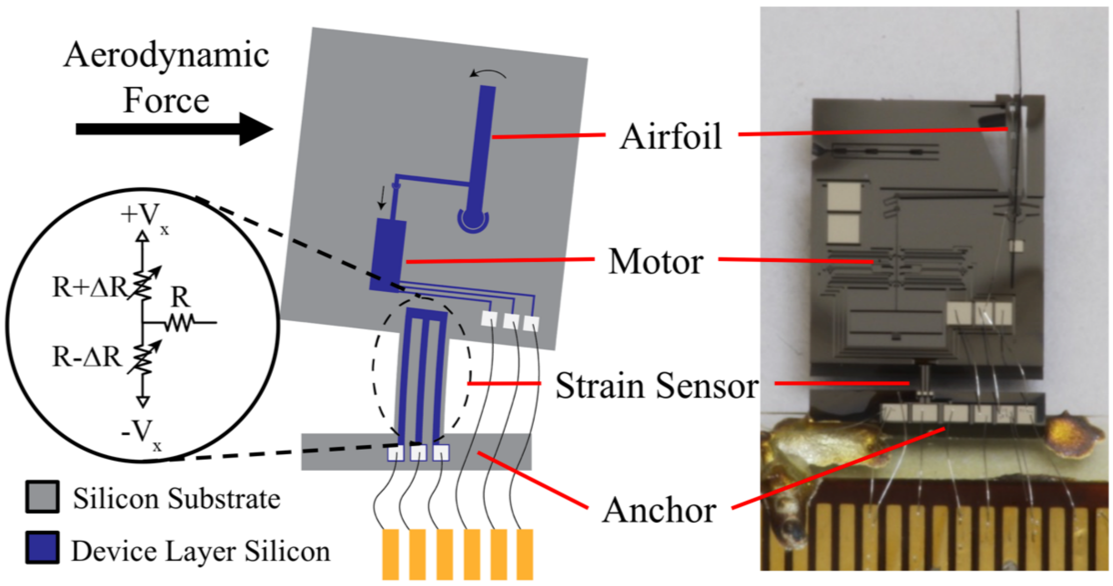


Figure 2.20: The strain sensor consists of a substrate layer cantilever with device layer piezoresistors patterned on top of the cantilever. When the cantilever deflects, it induces a differential strain in the piezoresistors on the edges of the cantilever. The anchor of the strain sensor is glued to a mount, so the rest of the device is suspended. When the airfoil rotates in airflow, it generates an aerodynamic lift force that deflects the strain sensor.

2.8 Aerodynamic Performance

Aerodynamic Performance of Static Fin

The theoretical performance of the airfoil was compared to actual performance by an experiment where the strain sensor was manually rotated with an attached airfoil. The results of this experiment are in figure 2.21. Airfoil lift force depended linearly on angle of attack for small angles. Lift force was not quadratically related to airflow speed.

Active Actuator Aerodynamic Performance

The next experiment measured the amount of aerodynamic force the MEMS control surface could generate while running in airflow. Once placed in 23 m/s airflow, the airfoil was periodically actuated while the strain sensor measured lift force. Figure 2.22 shows the device setup in this experiment.

Low frequency drift was present in the measurement signal, which required signal processing to remove. Force measurements were extracted by comparing force measurements at times when the fin was actuated to times when the fin was at rest (Figure 2.23). The measured periodic force signature correlated with fin deflection only occurs when both the fin is moving and the control surface is in 23 m/s of airflow, showing that the force measurement is due to

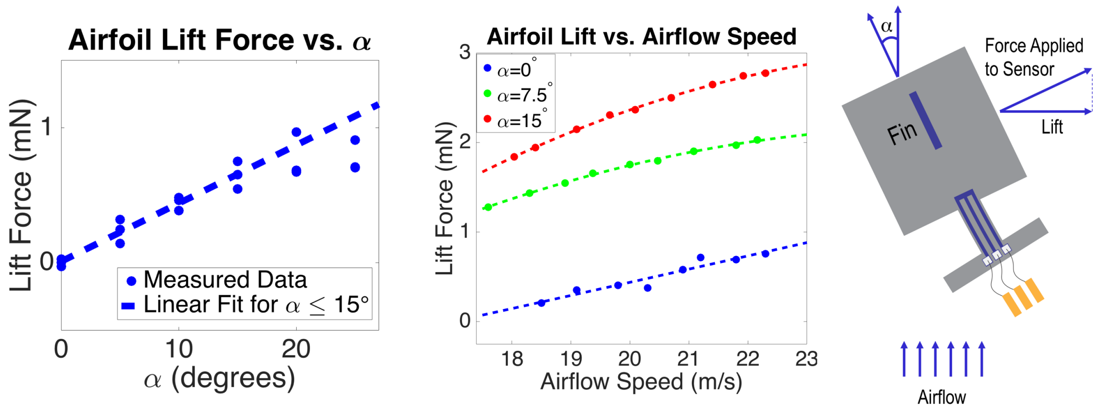


Figure 2.21: Top: Measured relationship between lift force, angle of attack, and airspeed for a 2x4 mm2 silicon airfoil. In left graph, experiments were run at 15 m/s flow speed. Bottom: a device with an immobilized airfoil in a wind tunnel was rotated while measuring the lift force.

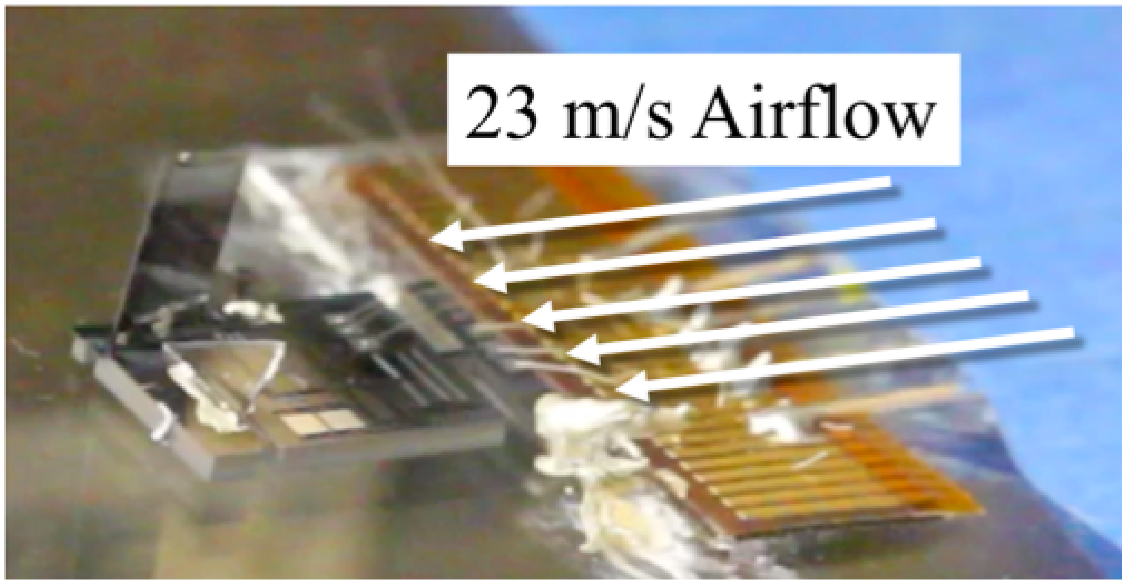


Figure 2.22: Image of the MEMS control surface system mounted for aerodynamic force experiments.

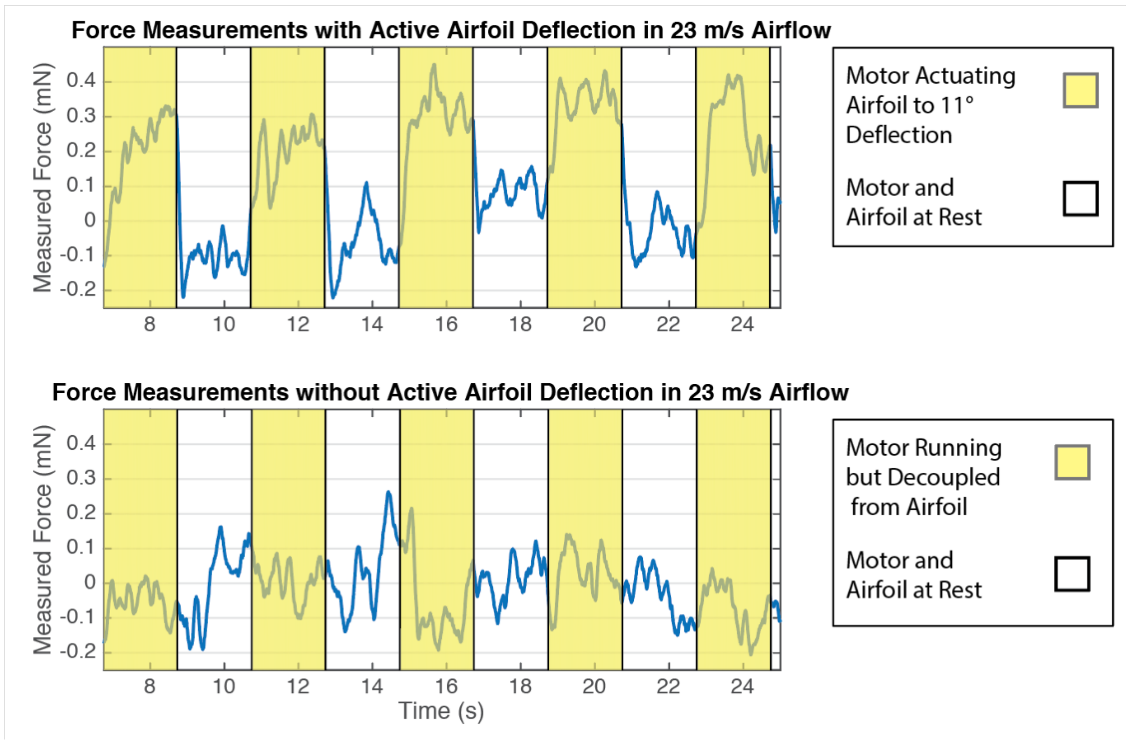


Figure 2.23: Aerodynamic performance graph. Top: Lift force generated by a periodically actuated airfoil. Noticeable force spikes occur when the fin is rotated by 11 degrees. Bottom: Control experiment showing force exerted on strain sensor when no actuation is occurring in 23 m/s airflow. In this control, the motor is running but is decoupled from the airfoil, so the airfoil doesn't deflect. In both plots, the 4 second moving average was subtracted from the signal

aerodynamic force generated by the fin. The average measured lift force was 0.25 mN, which was less than the predicted amount.

2.9 Systems Integration

Integrated Rocket System

These MEMS control surfaces will be used to create a pencil-sized autonomous rockets. In order to do that, we developed a process to assemble four control surfaces using standard wirebonding techniques and flexible printed circuit boards (PCBs) (Figure 2.24). Once the control surfaces are mounted on the PCB, it is wrapped around a custom fuselage fabricated by an inkjet 3D printer (Stratasys Objet260 Connex3).

The inchworm motors require 45V – 110V to operate, which poses a power supply challenge at small scales. We designed a portable 90V power supply system using a commercially-

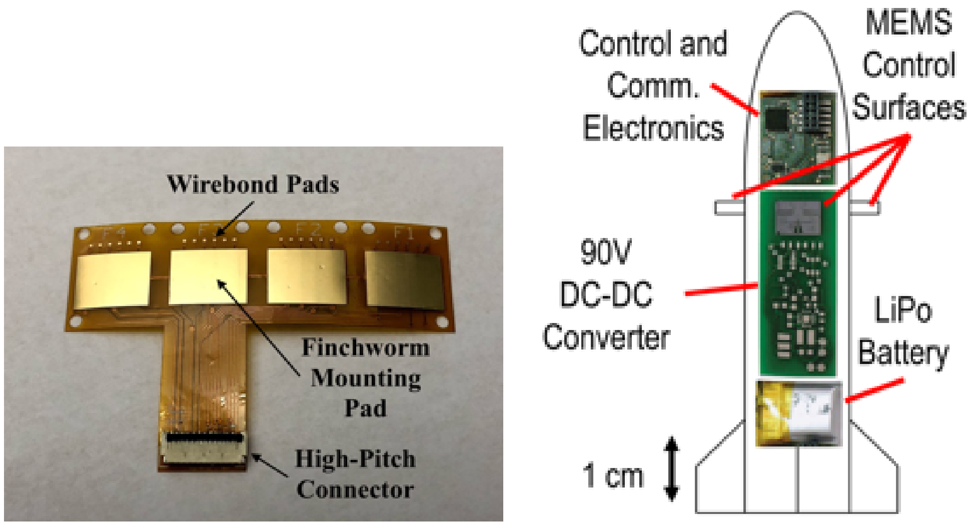


Figure 2.24: Integrated rocket system components. Top: Flexible PCB designed for assembling and routing control and power signals for the control surface system. The PCB is wrapped around a 3D-printed MEMS control surface fuselage module. Bottom: All of the electronics necessary to utilize the MEMS control surfaces has been demonstrated in a 1.5 cm diameter body tube.

available boost-converting integrated circuit (Linear Technologies LT4382), which has a 3mm x 3mm footprint. This system has demonstrated the ability to run a control surface for an hour on a 40 mAh, 3.7 V lithium ion battery. This board was 3.8 cm x 1.5 cm and weighed 2 g. Figure 2.25) shows an assembled pencil sized rocket including one control surface, battery, and control electronics.

Island Buckle Joint for Out-of-Plane Stability

Preventing out-of-plane forces from dislocating the silicon mechanisms requires manually glueing a grid of device-layer silicon on top of the rotary pin joint and the securing it with epoxy. This process is laborious and can potentially damage the device beyond repair. In order to avoid this assembly step, we designed a integrated SOI mechanism that prevents out-of-plane dislocation. This mechanism replaces rotary joint in the control surface. Figure 2.26 illustrates the operation of this island buckle joint. Inchworm motors were able to rotate the joint to half of the rotation range of the original rotary joint.

2.10 Towards Full System Integration

While steps towards integrating the control surface with a full autonomous rocket systems have been made, further development was required in order to enable this. Primarily, an

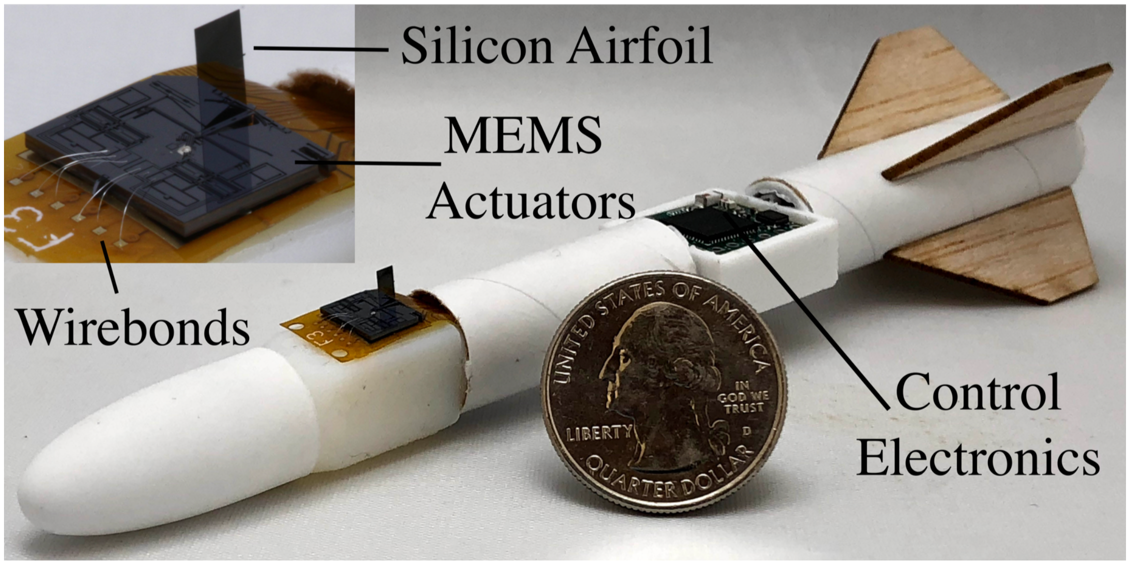


Figure 2.25: Pencil-sized rocket with one assembled MEMS control surface, LiPo battery, and control electronics attached

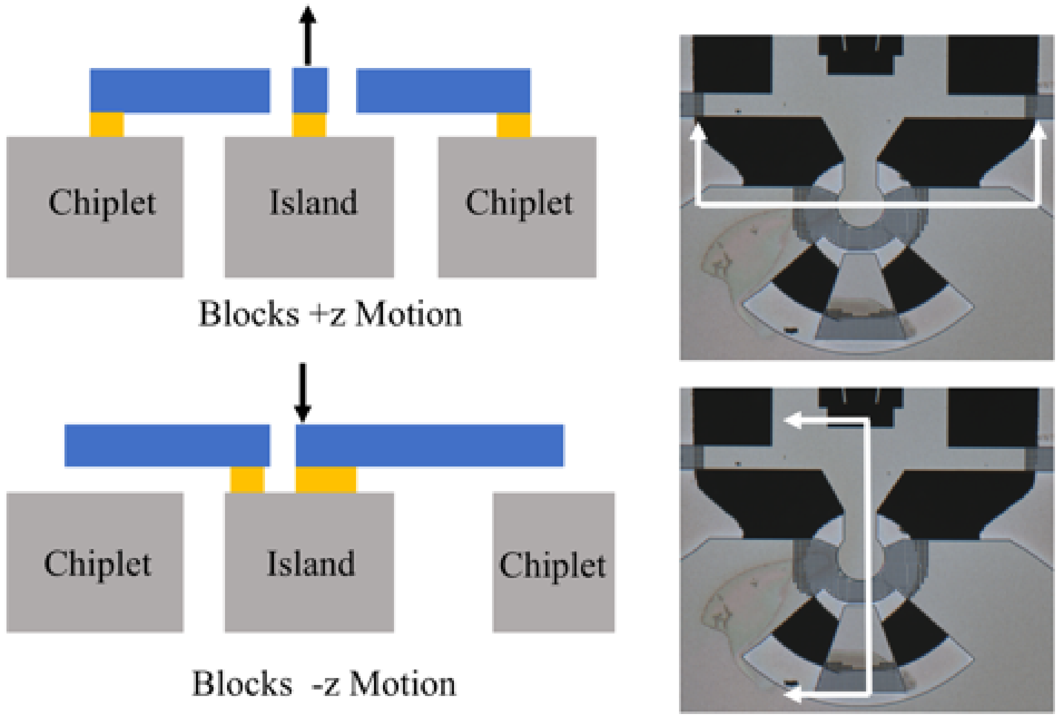


Figure 2.26: Cross-sections of an island buckle joint illustrate how it prevents out-of-plane dislocations of the device-layer mechanisms. Movement in the positive z direction is constrained by chiplet-anchored cantilevers, while movement in the negative z direction is contained by island-anchored cantilevers.

improved assembly process and platform for the MEMS control surface is required. The current assembly process for the control surface, which involves fastening the device onto a glass cover slip with cyanoacrylic glue, yields an assembly that is too large and cumbersome to integrate into a millimeter-scale rocket. Chapter 3 will discuss the improvements and developments required to enable an autonomous millimeter-scale rocket system.

Chapter 3

Integrated Rocket with Control Surface[1]

3.1 Intermediate Rocket System

While the ultimate target for the rocket system is the cigarette-sized system described in Chapter 1, an intermediate 20 cm rocket system is a more feasible short term goal for demonstrating a MEMS-actuated rocket. This system is based on the Micro Inertial Measurement System (MIMSY), which is a 1.6 x 1.6 cm^2 system that includes a Texas Instruments CC2538 wireless SoC, an IEEE 802.15.4 radio, and an MPU9250 inertial measurement unit (IMU) [40]. The inchworm motor power supply is a 80 V boost converter system based on Analog Device's LT3482 DC/DC boost converter. A 3.7V LiPo battery is used to provide power to the boost converter and the MIMSY.

[1]This chapter is adapted from [39]

System	**Component**	**Dimensions (mm^3)**
Propulsion	Estes A3-4T	13x13x45
Actuation	MEMS Control Surfaces	5x9x0.5
Power Electronics	ADI LT3482 Boost Converter	13x13x3
Power Supply	LiPo Battery	10x4x20
Computation, Communication, and Inertial Sensing	MIMSY	16x16x3
Total		13x13x200

Table 3.1: Summary of 20 cm Rocket System Components

3.2 Enabling the Integrated Rocket System

While the inchworm control surface was demonstrated to operate in a wind tunnel (Figure 2.22), the surface still had yet to be installed into this miniature rocket platform. The main obstacle of system integration was the assembly proces. The previous methods of assembly, gluing devices onto glass cover slides and the PCB lead wirebonds, were not suitable for a miniature rocket form factor. The glass assembly slide was fragile and too large to fit onto a rocket. It would also require a large number of individual assemblies and an intractable number of wires to harness. The solution to this was to develop a flex PCB that would aid mounting and assembly of multiple MEMS control surfaces. Figure 3.1 Shows the bare PCB used for these. This PCB provides wirebond pads for the control surfaces. During assembly, the control surfaces are glued with silver epoxy onto the large ground pad, which grounds the substrate of the device. Devices with grounded substrates have been observed to operate more effectively. Wirebonds are used to electrically connect the inchworm motors with to the flex PCB. The reliability of this wirebonding step was improved by epoxying a 0.2 mm thick carbon fiber sheet (ACP Composites SKU: CFL-TW-0.008-ADH-1) onto the backside of the flex PCB, underneath the wirebond and grounding pads. This increases the stiffness of the board which makes the wirebonding power transfer better and increases durability to the board bending during assembly. This could also be solved by specifying flex stiffener in these areas during PCB design, but the fabrication facility used for this board does not provide flex stiffener. Figure 3.1 shows a fully assembled PCB with multiple devices. Multiple devices can be assembled at once since the board lies flat on the wirebonding surface. Once assembled, this flex PCB can be wrapped around the fuselage of a rocket. The control surface PCB assembly was connected to the MIMSY via header pins soldered to the connector pads of the flex PCB.

After assembly, the flex PCB can be wrapped around a fuselage component of the rocket, which mounts the control surfaces onto the rocket. This allows the flex PCB to be fully assembled with four control surfaces while the PCB is flat before it is mounted onto the rocket. Otherwise, each device would have to be individually mounted onto the rocket which would increase the difficulty of manufacture. Figure 3.2 illustrates a flex PCB mounted onto a rocket fuselage component.

Another difficult aspect of assembling the device in [38] was adding the out-of-plane protection bars. One of the primary goals of the next design revision in this chapter was to address this manufacturing difficulty.

Another goal of the following design revision was to increase the amount of force the actuator could generate, which would improve the maximum lift force the control surfaces could generate. This in turn would improve the maneuverability of the rocket, which is especially important in the 10 cm rocket case as it's larger, heavier, and thus more difficult to maneuver than the target cigarette-sized microrocket. Stronger inchworm motors are required to increase the maximum lift force supported by the control surfaces. In the next design iteration, the inchworm motors were optimized in the same method as [13, 56] in order to increase the force density and layout efficiency of the motors.

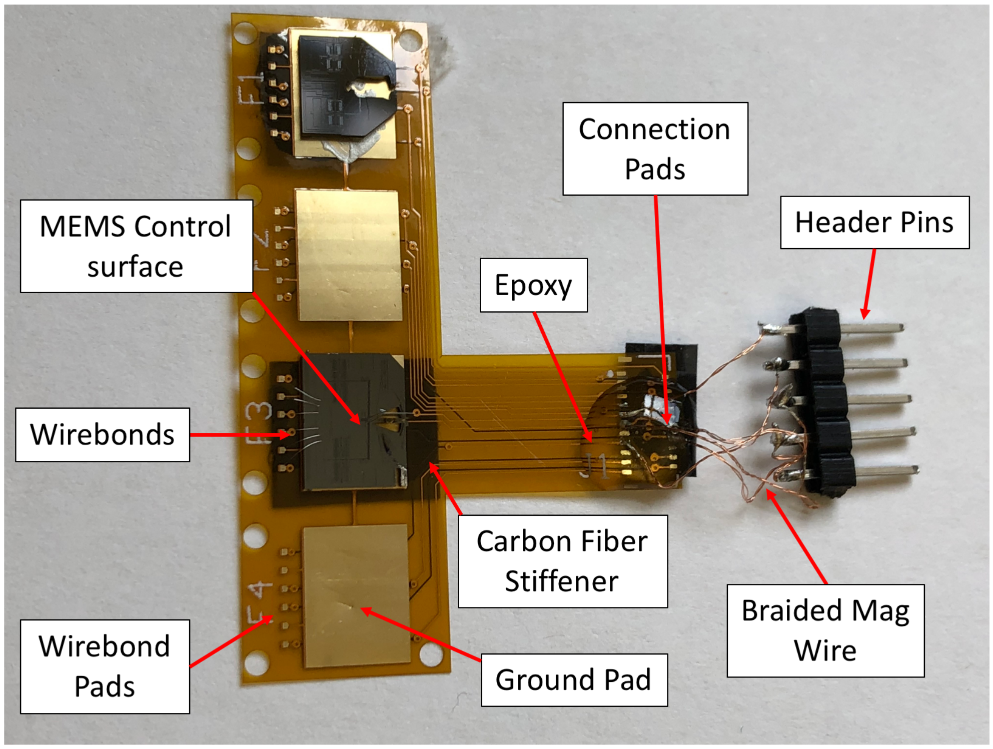


Figure 3.1: Annotated image of MEMS control surface flex PCB. Four control surfaces can be mounted onto this board at onfcce. This flex PCB has stiffener underneath the wirebond and ground pads, which keeps the wirebond surface and substrate bonding surface flat. Manufacturer-standard flex stiffener could also be used as a stiffener for these pads. The header pins are soldered onto the connection pads and then are encased in epoxy to increase their structural strength.

Implementing these changes, we were able to integrate the MEMS actuator into a fully contained 16.9 g, 20 cm rocket system, which is composed of a battery, wireless sensor node, IMU, and high voltage buffer board for running the inchworm motors at 80 V. The following chapter describes this integration and demonstrates the ability of the MEMS control surface to induce a roll maneuver in a wind tunnel.

3.3 Theory and Design

Aerodynamics and Airfoil Design

Most aircraft use control surfaces such as ailerons, elevons, and canards to control their trajectories. For small angles of attack, ($\alpha <15°$), the lift force F_l produced by these airfoils can be modeled using thin airfoil theory [55].

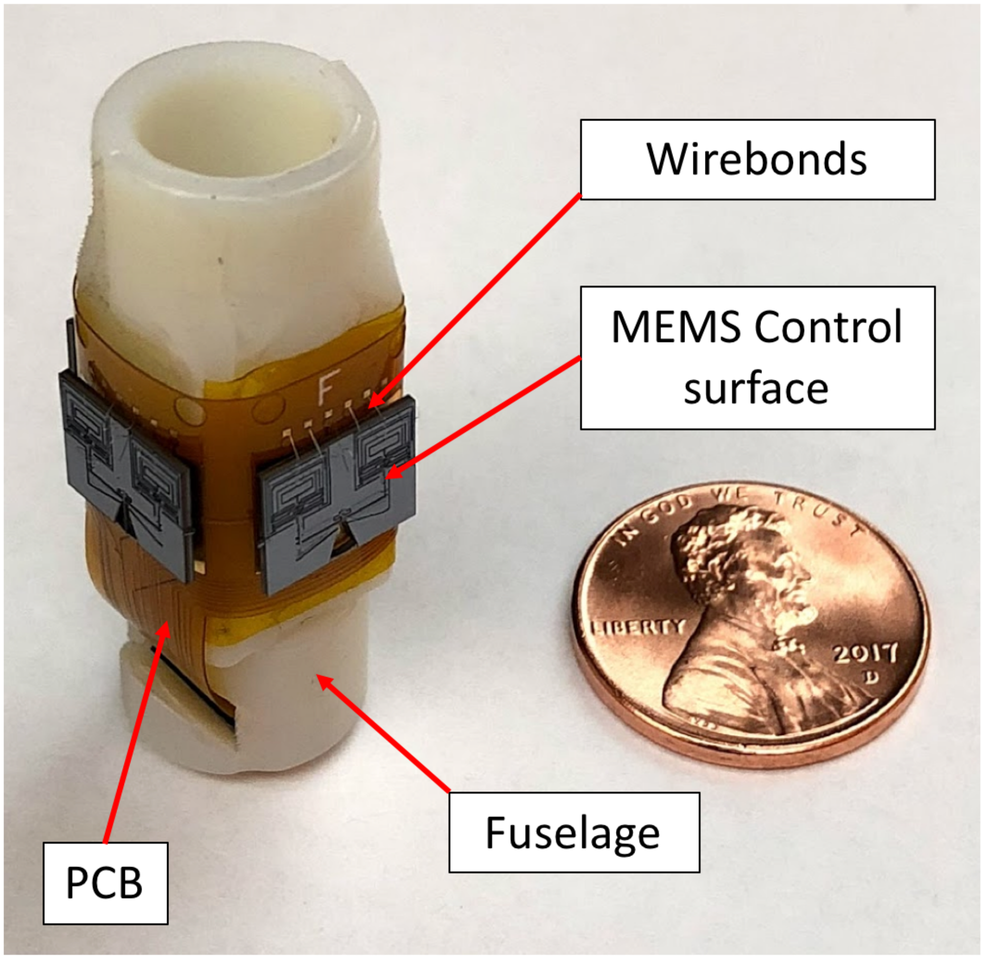


Figure 3.2: Example of an assembled control surface PCB mounted on a rocket fuselage module.

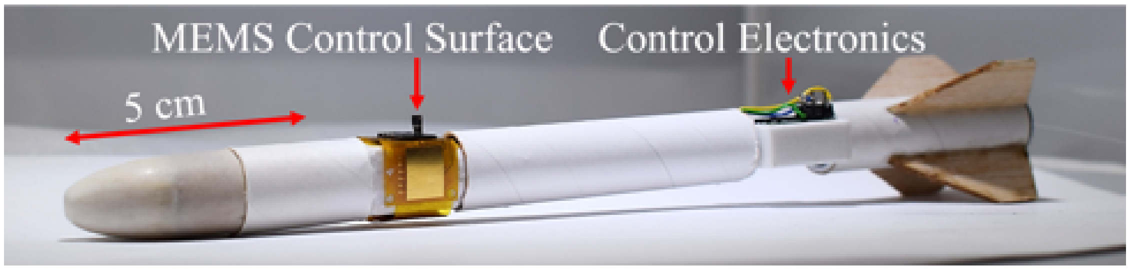


Figure 3.3: The fully assembled rocket, consisting of a MEMS control surface and its 2x4 mm2 steel airfoil, a single-cell LiPo battery, and control electronics.

$$F_l = \frac{1}{2}\rho v^2 2\pi\alpha A \tag{3.1}$$

where ρ is the density of air, v is the airflow velocity, and A is the airfoil's area. Our control surface uses electrostatic inchworm motors and rotary pin joints to rotate a 2x4x0.03 mm^3 steel foil airfoil. Previous simulations showed that control surfaces generating 10 mN of lift force suffice to control a millimeter-scale rocket [2]. With our airfoil's dimensions, we can achieve this lift force at an airflow velocity of $v = 35$ m/s, and we've previously observed in simulation that our rocket can readily achieve this speed throughout most of its flight path [9]. To support this amount of lift force, the inchworm motor actuator from [38] will need to be redesigned to produce more force. The resulting aerodynamic lift force applies a torque on the rocket, providing us with a method of controlling roll and yaw.

$$\tau_{roll} = F_l(r + r_{CoP}) \tag{3.2}$$

$$\tau_{yaw} = F_l d_{CoP} \tag{3.3}$$

where r is the rocket's body radius (7.5 mm), $r_C oP$ is the radial distance from the base of the fin to the fin's center of pressure, and d_{CoP} is the distance between the rocket's center of gravity and the fin's center of pressure (defined by the fin's quarter-chord within thin airfoil theory). Equations 3.2 and 3.3 allow us to control roll and yaw with one aerodynamic airfoil, and adding multiple airfoils around the rocket's circumference gives us full control over roll, pitch, and yaw.

MEMS Actuator Mechanism

The MEMS actuator, fabricated in a simple 2-mask silicon-on-insulator (SOI) process [52, 14], comprises a pair of electrostatic inchworm motors and rotary pin joints. The inchworm motors pull on a lever arm attached to the airfoil slot, which rotates the airfoil (Figure 3.4). We applied the higher-density electrostatic inchworm motor design described in [11] to increase our force density by 65 percent compared to previous work done in [10], as measured by the ratio between the capacitive finger overlap area and the entire motor's layout area. This redesign allowed us to shrink the MEMS device's area by 21 perent from 9x7 mm^2 to 5x9 mm^2, realizing our design goal of a MEMS actuator that can fit on a 6 mm diameter rocket [9]. Previous work has attempted to prevent out-of-plane forces from dislocating the silicon mechanisms by gluing a small silicon piece over the rotary pin joints [10]. One problem we found with this process was that commercial silver epoxy or superglue would often wick a significant distance from its initial application position, immobilizing silicon mechanisms. To solve this issue, we added extra slots to either side of the MEMS device, where we slotted in a bracket that lay over the lever arm and pin joint structures (Figure 3.5). The bracket is secured via epoxy only at the sides to prevent wicking from affecting any mobile MEMS components. We fabricated the brackets using the same 2-mask SOI process, and we observed no degradation in device performance after placing the bracket.

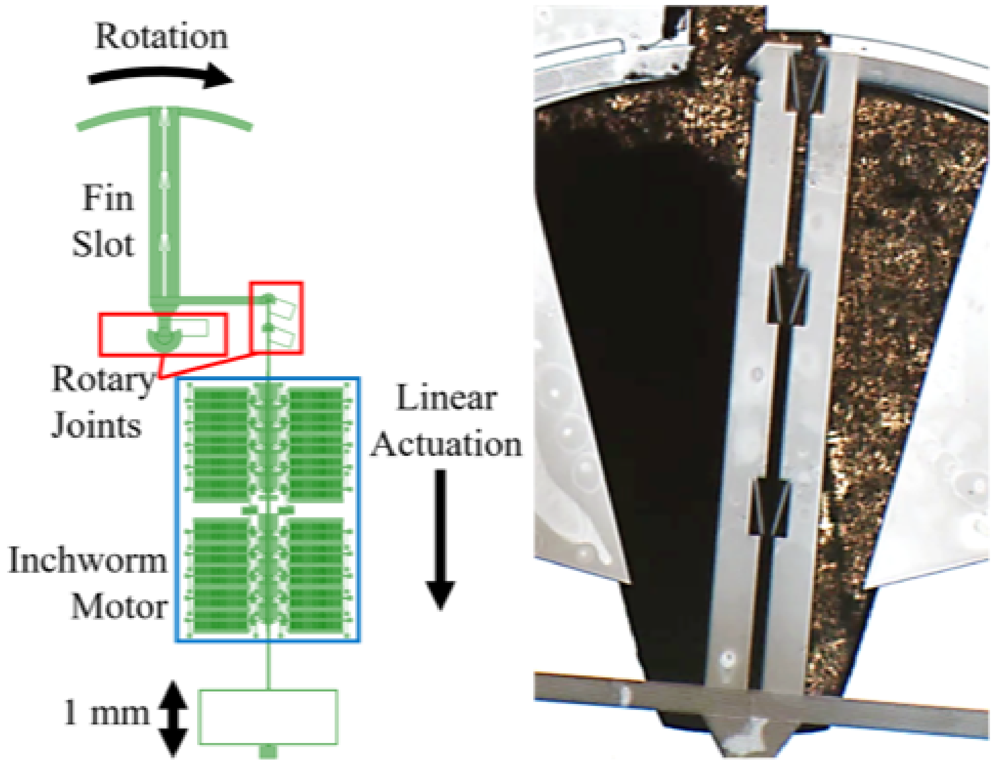


Figure 3.4: (Left) Schematic of half of the MEMS actuator mechanism. The inchworm motor linearly actuates the rotary pin joints, which rotate the airfoil slot. (Right) A microscopic photograph of the airfoil slot after rotation.

3.4 Integrated Rocket Design

MEMS Actuator Mechanism

To mount the assembled control surfaces onto the rocket, we used the custom flexible printed circuit board described in Figure 3.1. The fully assembled flex PCB in this application is shown in Figure 3.5. The MEMS device substrate is electrically connected via silver epoxy to a large ground plane, and the other control signals are routed from the PCB to the control surface via standard wire bonding techniques. The assembled flexible PCB is then wrapped around a custom 3D-printed fuselage and friction fit into the rocket body (Figure 3.3).

One problem we found in practice with this process was that any curvature in the flexible PCB or flexibility in the substrate substantially decreased our wire bonding yield. To solve this, we attached a 0.2 mm thick carbon fiber stiffener behind the ground plane before the epoxy step, which greatly improved our wire bonding yield. Our placement of the airfoil closer to the rocket's front end follows from the design of canard guided missiles, which feature motorized frontal control surfaces and passively stabilizing tail fins. By shifting the

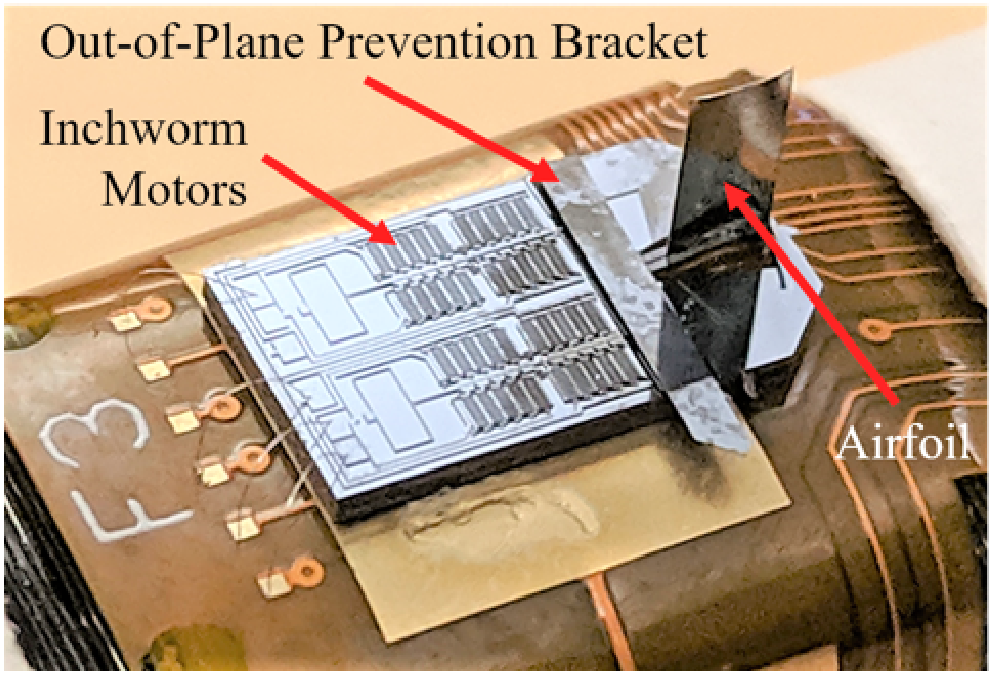


Figure 3.5: The assembled MEMS control surface.

rocket's center of pressure forward, this design model improves the rocket's response time and control surface output gain, reducing the required control surface hinge moment compared to equivalent tail fin controllers and decreasing the rocket's miss distance against mobile targets [12]. The design also facilitates rocket assembly by keeping the control surfaces and electronics far from the rocket engine.

Electronics and Control

The inchworm motors require 45 V–110 V to operate, which poses a power supply challenge at small scales. To solve this, we developed an 80 V high voltage buffer board based on Linear Technology's LT3482 DC/DC boost converter. The board's 13 mm diameter footprint enables it to be mounted normal to the rocket's body axis, presenting a 4x area decrease and a 5 cm rocket length reduction compared to previous iterations [10] (Figure 3.6). The rocket's onboard control and wireless communication are handled by a MIMSY [13]. Input power was supplied via a 40 mAh single-cell lithium polymer battery, which can power the IMU and a constantly running TX radio broadcast for roughly an hour after the rocket lands. Figure 3.7 shows the block diagram of the electronics systems.

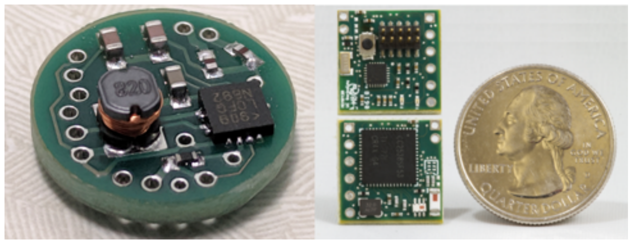


Figure 3.6: (Left) The 13 mm diameter high voltage buffer board. (Right) The MIMSY control and wireless node.

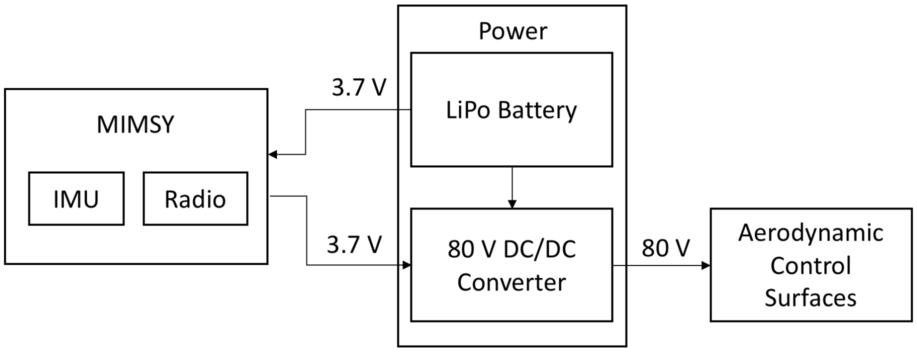


Figure 3.7: Block Diagram of Rocket Electronics System

3.5 Aerodynamic Performance

The system was tested inside a wind tunnel with 13.3 m/s airflow. Because the rocket's moment of inertia about its roll axis is much smaller than about its pitch or yaw axes, for ease of measurement we focused on measuring the roll generated from the airfoil's aerodynamic lift. The full testing setup is shown in Figure 3.9. The rocket's vertical suspension by tensioned string in the wind tunnel allowed us to model the system as a torsional pendulum.

$$\Sigma\tau = I\dot{\Theta} = \tau_{roll} - K_{torsion}\theta \tag{3.4}$$

$$K_{torsion} = \frac{GJ}{L} \tag{3.5}$$

where I the moment of inertia along the roll axis, $K_{torsion}$ is the suspension string's torsional coefficient, G is the string's shear modulus, J is the string's second moment of area ($J = \pi r^4/2$ for a circular cross-section), and L is the string's length. We used 0.005" diameter Berkley XLPS2-15 fishing line for our suspension string. To characterize the fishing line's shear modulus, we measured its period of oscillation as a torsional pendulum given a test load of known moment of inertia. We fitted this data to Equation 3.5 to show that $G = 6.9$ GPa for our fishing line with an r^2 value of 0.96, indicating high convergence to the theory. To measure the control surface's output torque τ_{roll}, we measured the rocket's

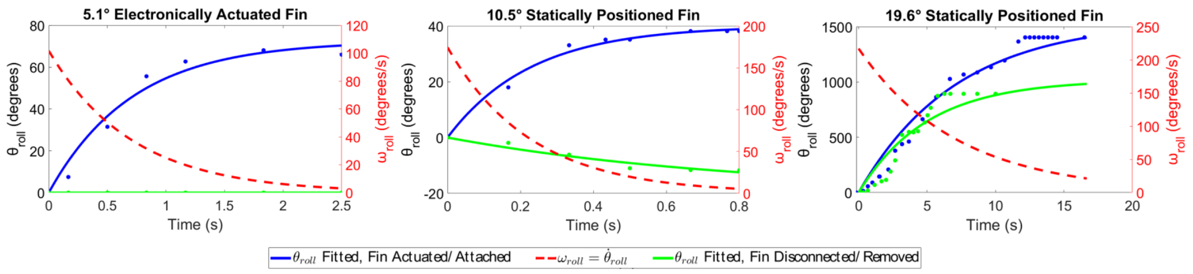


Figure 3.8: Roll angle vs. time data for individual rocket experiments in the wind tunnel, measured visually. The plots show the difference between the rocket's roll with and without an actuated fin. From left to right, the experiments were conducted at $K_{torsion} = 2.6\mu$Nm, 4.9μNm, and 0.51μNm. The 5.1° and 10.5° experiments were conducted in 13.3 m/s airflow while the 19.6° experiment was conducted in 20.1 m/s airflow, which explains the latter's large roll angles.

equilibrium position about its roll axis at different wind speeds and angles of attack (Figure 3.8). We tested fins at three different angles of attack: the 5.1° fin is electronically actuated, while the 10.5° and 19.6° fins are glued into place as control experiments. To account for any asymmetry in the rocket's construction, we measured the difference between the rocket's roll with and without the battery connected (or, in the case of the statically positioned fins, by replacing the MEMS actuator with a finless one). We also measured the torsional coefficient and airflow speed for each setup. Table 3.10 factors in all these measurements to calculate the output torque on the rocket body. Experimental airfoil output torque is shown to match closely with the theory (Equation 3.2) for angles of attack under 10°, and the deviation between 10-20° angle of attack also matches the expected limit of thin airfoil theory from nonidealities. These results present a promising case for our ability to use linear control models to accurately control the rocket during flight.

We simulated the rocket's launch performance with an Estes A3-4T engine using the software OpenRocket [14]. The rocket should have a maximum range of 360 m, a top speed of 79 m/s, and the ability to execute 5.5g maneuvers.

3.6 Future Vision

The eventual design of a cigarette-sized rocket would comprise the Single-Chip Micro Mote (SCμM) [6] for communication and computation, an IMU for navigation, MEMS control surfaces for control, a miniaturized camera for sensing, and a rocket motor like in [5] for propulsion. The high voltage circuitry will be replaced by millimeter-scale high-voltage DC/DC converter integrated circuitry [15]. Power will be supplied via two 1.5 V coin cell batteries like the SR521SW. The rocket would be 10 cm long, 6 mm in diameter, and would have 400 m range, a top speed of 25 m/s, and the ability to execute 10g manuevers [9]. These

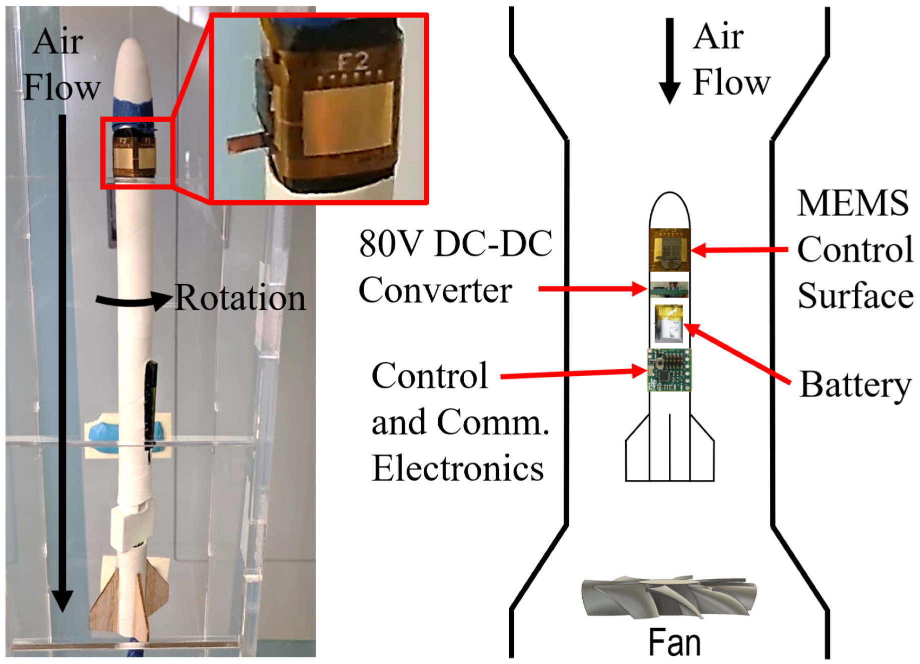
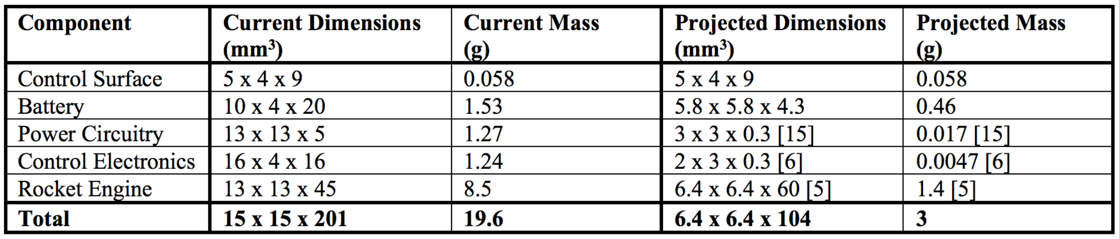


Figure 3.9: (Left) Lift force is generated when the fin is rotated, causing the entire rocket to roll along its axis. (Right) Schematic of the integrated rocket system inside a wind tunnel.

Angle of Attack (°)	$\tau_{roll,ideal}$ **(μNm)**	$\tau_{roll,meas}$ **(μNm)**	**σ (μNm) (N=3 trials)**
5.1[1]	4.2	3.6	0.035
10.5[2]	8.6	7.8	0.23
19.6[2]	15.8	7.8	0.13

Figure 3.10: Output torque from the fin's lift force, measured across 3 trials for each angle of attack. All measurements were normalized to match a 2x4 mm^2 airfoil's area and a wind speed of 13.3 m/s. The rockets were either for 1. electronically actuated or 2. statically positioned systems.



Component	**Current Dimensions (mm^3)**	**Current Mass (g)**	**Projected Dimensions (mm^3)**	**Projected Mass (g)**
Control Surface	5 x 4 x 9	0.058	5 x 4 x 9	0.058
Battery	10 x 4 x 20	1.53	5.8 x 5.8 x 4.3	0.46
Power Circuitry	13 x 13 x 5	1.27	3 x 3 x 0.3 [15]	0.017 [15]
Control Electronics	16 x 4 x 16	1.24	2 x 3 x 0.3 [6]	0.0047 [6]
Rocket Engine	13 x 13 x 45	8.5	6.4 x 6.4 x 60 [5]	1.4 [5]
Total	**15 x 15 x 201**	**19.6**	**6.4 x 6.4 x 104**	**3**

Figure 3.11: Current and projected rocket volumetric dimensions and mass with future scaling of rocket components. The control surfaces presented in this paper are well suited for miniature rockets as is, and incorporating the referenced circuitry and rocket engine designs could further reduce rocket size. The total row includes the above factors plus the rocket's nose cone, electrical wiring, and fuselage.

improvements are summarized in Table 3.11.

3.7 Conclusion

We demonstrated the integration of MEMS aerodynamic control surfaces into a fully integrated miniature rocket. Improvements in assembly techniques and MEMS design substantially improved yield, helping to create MEMS aerodynamic actuators ready for use in real-world systems. These results present a promising case for the ability of MEMS aerodynamic control surfaces to sufficiently control miniature rockets during actual launches, and our higher force density actuators indicate future potential for continued size reduction.

Part II

Lighthouse Localization for Autonomous Microrobotic Systems

Chapter 4

Lighthouse Localization of Single Chip Micro Mote

4.1 Introduction

Lighthouse localization was initially conceived for localizing Smart Dust [62, 63]. More recently, it has become a successful and accurate 3D localization technology used in virtual reality and industrial metrology, providing millimeter-scale accuracy [63, 27, 28, 64]. This technology relies on a “lighthouse beacon” that generates a series of wide-angle infrared pulses and precise planar laser scans. These pulses and scans are detected by the localizing motes (small sensor and processing platforms) and their timings are used by the motes to calculate the motes’ azimuth and elevation relative to the lighthouse beacon (Figure 4.5). Lighthouse localization provides unique advantages for localizing large numbers of miniaturized, low power, wireless motes. First, this method scales well computationally with large numbers of motes, because the localization time is independent of the number of motes being localized. Second, the distributed localization computation required by the motes to process the lighthouse measurements is cheap, which is suitable for a miniature low-power microcontroller. Additionally, the hardware required to detect lighthouse beacon pulses requires little power. Finally, the motes don’t require RF communication in order to localize themselves, which frees radio bandwidth for other uses. When coupled with miniaturized monolithic integrated systems, lighthouse localization could enable of host of applications such as fine granularity wireless sensing, body-area networks, and multi-agent control and localization of microrobots.

Autonomous microsystems, such as microrobots, have limited payload capabilities, which require their processing and communication platforms to be highly miniaturized. The $2 \times 3 \times 0.3$ mm^3, 5 mg, Single Chip Micro Mote (SCμM) [25] is an example of a suitable system for microrobots, containing an ARM Cortex-M0 microprocessor and a 2.4 GHz radio compatible with IEEE 802.15.4 and Bluetooth Low Energy. Localization of these systems is important for them to perform intelligent tasks in the real world. Lighthouse localization is a

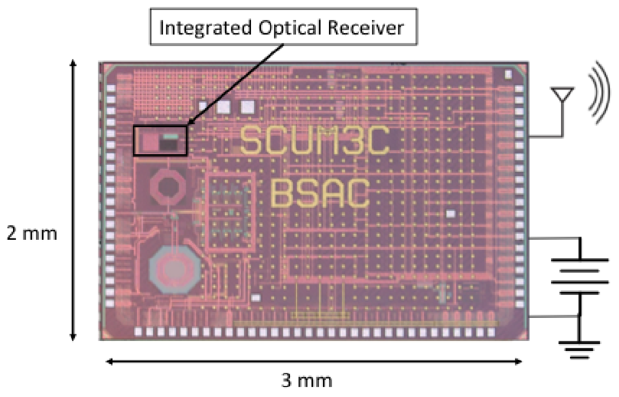


Figure 4.1: Die photo of the Single Chip Micro Mote (SCμM). The integrated optical receiver is initially designed for contact-free optical programming. In this paper, we re-purpose it to allow the mote to localize itself by timing the horizontal and vertical laser scans from a HTC Vive lighthouse localization system.

promising system for localizing SCμM.

The tracked object in the lighthouse system requires an infrared-sensitive photodiode that detects the lighthouse station's laser sweeps. In [65], we demonstrated that SCμM's integrated optical receiver, originally intended for contact-free optical programming, can be repurposed to accurately detect the laser sweeps generated by an HTC Vive lighthouse base station[1], which enables calculation of the mote's azimuth and elevation angles relative to the lighthouse. This work further develops this application and demonstrates the full 3D tracking of a SCμM chip, while using its on-board radio to wirelessly communicate its position.

4.2 The Single Chip Micro Mote (SCμM)

The Single Chip Micro Mote (SCμM), shown in Fig. 4.1, is a fully monolithic wireless System-on-a-Chip developed for microbotic applications [25]. It contains a crystal-free 2.4 GHz radio that is compatible with the IEEE 802.15.4 standard and has limited Bluetooth Low Energy transmit capability, an ARM Cortex-M0 processor with 64 kB of program and data memory, and an integrated optical receiver for contact-free programming [65]. Intended for payload-constrained applications like micro-robotics, the $2\times3\times0.3$ mm^3 SCμM chip does not require any external connections except for a power source and an antenna. For example, SCμM has been demonstrated driving an electrostatic inchworm-motor-powered microrobotic gripper [41]. Additionally, Chang et al. successfully demonstrated SCμM running a low-power time-synchronized network protocol (6TiSCH) and participating in a time-synchronized channel hopping mesh network [66].

[1] `https://www.vive.com/eu/accessory/base-station/`

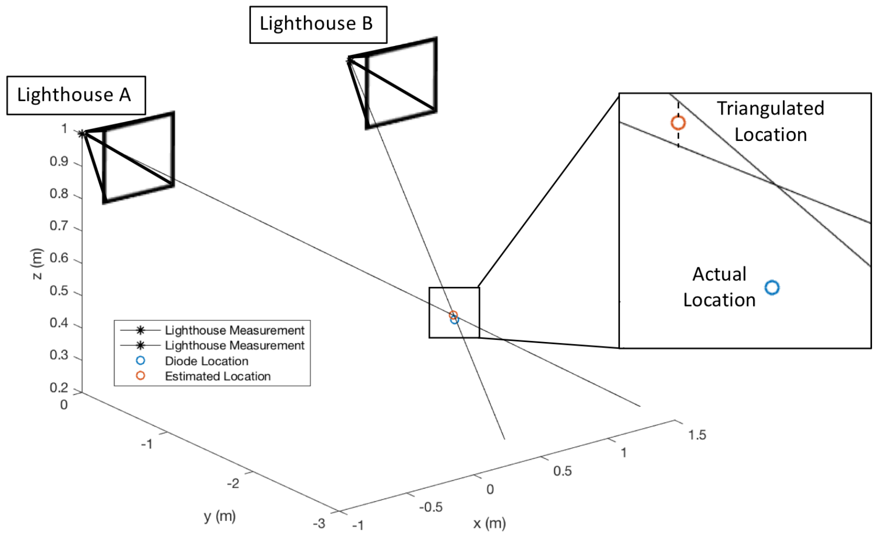


Figure 4.2: Triangulation of lighthouse measurement rays. The relative angle measurements from each lighthouse create rays pointing in the direction of the mote. These projected rays are unlikely to cross due to measurement noise, so the direct linear transform is used to estimate their intersection and triangulate the SCμM's position.

Localizing SCµM is an important ability, as it could enable the localization of microrobots and sensor networks that utilize SCµM. However, the computational ability of SCµM is heavily limited by it size and power constraints. Additionally, the monolithic nature of SCµM reduces the utility of localization methods that require large additional hardware (e.g ultrawideband localization, ultrasonic, gps etc.). In these cases, the hardware required for localization is significantly larger than SCµM, which defeats the purpose of having SCμM in the first place.

Fortunately, SCµM has a integrated photodiode that can be used for localization, thus enabling 3D tracking without any additional SCµM-related hardware. The photodiode is a component of SCμM's integrated optical receiver, initially designed for contact-free optical programming. It comes with the necessary circuitry which, upon receiving a specific 32 bit bootload start symbol, copies the subsequent bitstream into memory, and resets the micro-controller. A special optical programming board is used to transfer a compiled binary from a computer onto SCμM (Figure 4.3). The active power of the optical receiver is 1.5 μW, compared to the active power of the entire system, which can be up to 2 mW [65]. While the integrated photodiode was originally designed for optically bootloading code onto SCµM (Figure 4.4), it is also able to detect temporally-structured light from the HTC Vive V1 lighthouse base stations, which are used in commercially-available virtual reality systems. This lighthouse localization system is well-suited for localizing SCµM due to it's light computational requirements and favorable scalability for large numbers of motes.

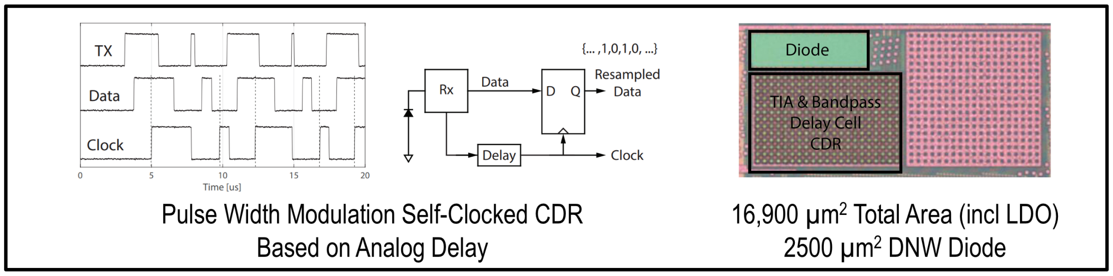

TX
Data
Clock
Time [us]
Rx
Data
D Q
Resampled Data
{..., 1,0,1,0, ...}
Delay
Clock
Diode
TIA & Bandpass Delay Cell CDR
Pulse Width Modulation Self-Clocked CDR Based on Analog Delay
16,900 µm² Total Area (incl LDO) 2500 µm² DNW Diode

Figure 4.3: SCμM receiver summary [65].

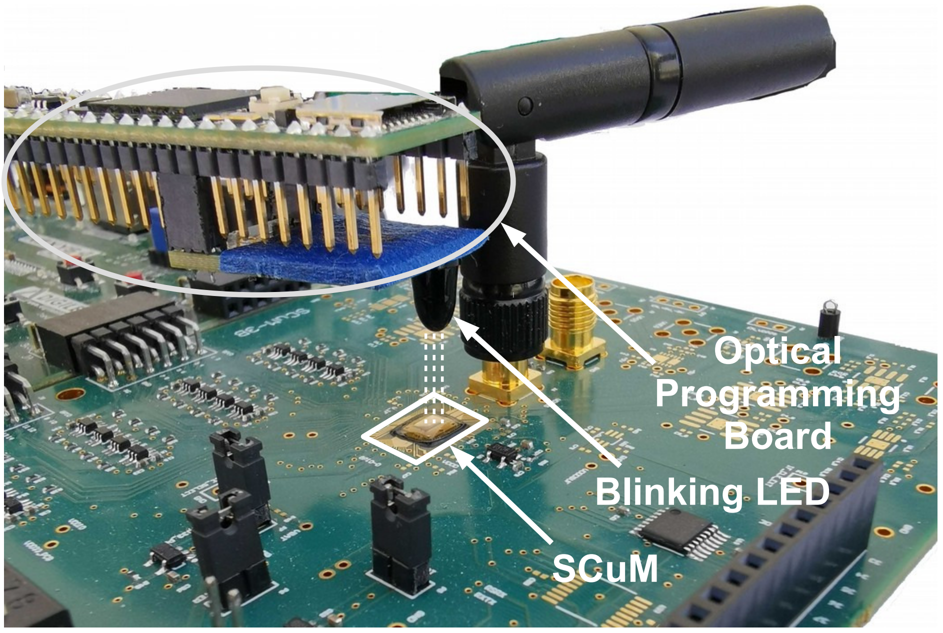


Figure 4.4: The optical programming process of SCμM (here shown soldered onto its development board). An optical programming board, connected to a laptop, blinks an LED: it starts by a bootload blink pattern, followed by the stream of bits contained in the binary image to load. Upon receiving the bootload blink pattern, dedicated circuitry in SCμM copies the subsequent bit stream into memory, then resets the micro-controller. The optical receiver is repurposed to time the flashes received from a lighthouse base station during the execution of the firmware loaded, to allow the SCμM chip to self-localize.

4.3 Lighthouse Localization

Base Principle

Lighthouse-based localization was initially developed for localizing constrained low-power electronic objects [63]. In recent years, it has been applied for precisely and quickly measuring the position and orientation of Virtual Reality (VR) headsets such as the HTC Vive [64].

Pulse Type	Duration	Description
T_A	62.5 μs	sync pulse announcing azimuth sweep
T_E	72.9 μs	sync pulse announcing elevation sweep
T_S	104.0 μs	sync pulse announcing skip (no sweep)
T_{sweep}	8.3 ms	full 180 degree sweep of the laser

Table 4.1: Duration of the pulse activity of an HTC Vive Lighthouse.

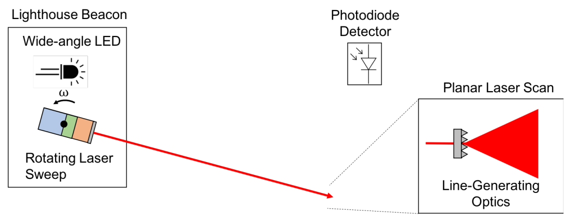
The base principle is that a lighthouse base station (which is typically the size of a coffee cup) is equipped with a laser which sweeps a laser plane across space, first horizontally, then vertically. An object equipped with a photodiode timestamps the moments it detects the pulse from the horizontal and vertical sweeps. Knowing the speed of those sweeps, it can compute its azimuth and elevation angles, and hence knows it is located on a ray relative to the lighthouse base station (Figure 4.5).

To obtain a 3D localization, at least two base station are needed, as depicted in Fig. 4.2. These lighthouses are connected by a wire which allows them to synchronize their pulses to one another; the role of each lighthouse (Lighthouse 1 or Lighthouse 2) is configured by a slider on each lighthouse. Fig. 4.6 shows a chronogram of the activity of each lighthouse. The lighthouses are equipped with two types of light sources: a high-powered omnidirectional infrared LED (used for sending sync pulses), and two lasers pointed at either of two mirrors that rotate at 120 Hz, at a 90 degree angle. The mirrors are used to sweep through space either in the azimuth or elevation, causing SCμM to receive a laser pulse. The lighthouses alternate between sending sync pulses and sweeping one of their lasers. Lighthouse 1 always sends a sync pulse before Lighthouse 2. The duration of the sync pulse indicates what the lighthouse will do, per Table 4.1: sweep its azimuth laser, sweep its elevation laser, or keep its lasers off. The duration before receiving the laser pulse encodes the angle to the lighthouse. After receiving an azimuth and elevation pulse, SCμM can compute on what measurement ray it is relative to the lighthouse that send the pulses. Because of measurement inaccuracies, the measurement rays from both lighthouse base station most likely don't intersect. The challenge is hence to compute the position of the point in 3D space which minimize the distance to both rays.

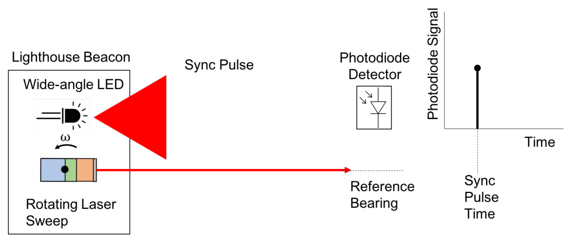
Receiving Lighthouse Laser Pulses on SCμM

As described in [65], SCμM is able to detect the pulse from an HTC Vive Lighthouse using its optical receiver. We develop custom firmware[2] to process the structured infrared light emitted by the base stations. This firmware uses interrupts to detect, measure, and decode the laser scan timings and sync pulse widths that are received by SCμM's optical receiver. The output of the optical receiver is routed to a GPIO output pin, then back into the processor via

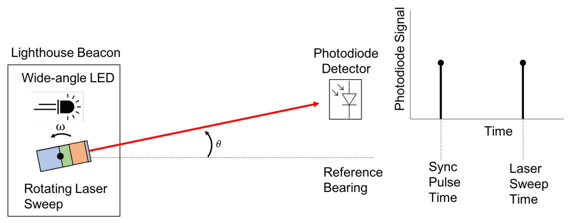
[2] available at `https://github.com/PisterLab/scum-test-code`.



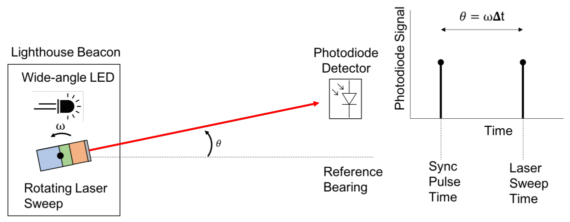
(a)



(b)



(c)



(d)

Figure 4.5: Illustration of lighthouse localization as implemented in the HTC Vive V1. a.) The system consists of a lighthouse beacon containing a wide-angle LED array, two sets of rotating planar lasers, and a photodetecting system. b.) As the the rotating laser scan passes a reference heading, the LED array flashes and generates a synchronization pulse, which is detected by the photodetector. c.) Once the planar laser scan reaches the photodetector, the photodetector detects the scan and measures the time difference between the sync pulse and the laser scan. d.) Using the constant rotation frequency of the laser scan and the time difference, the photodetector can calculate its heading relative to the reference heading of the lighthouse beacon.

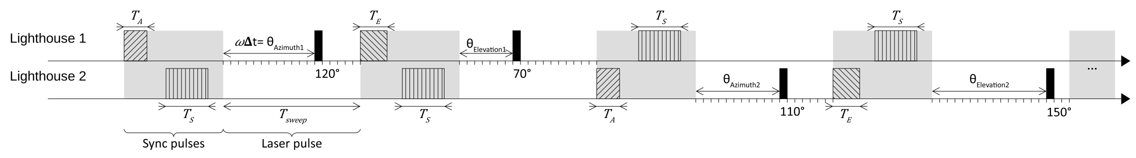


Figure 4.6: Chronogram of the pulses received by the SCμM chip from the two lighthouses. The lighthouses are synchronized over a wired interface. They alternate between periods when sending omnidirectional sync pulses, and periods when sending laser pulses. Lighthouse 1 always sends a sync pulse before Lighthouse 2. The duration of the sync pulse indicates what the lighthouse will do, per Table 4.1. The duration before receiving the laser pulse encodes the angle to the lighthouse. In this illustration, SCμM measures 120 degrees azimuth / 70 degrees elevation from Lighthouse 1, 110 degrees azimuth / 150 degrees elevation from Lighthouse 2.

multiple GPIO pins: one connected to an active high level interrupt and one connected to an active low level interrupt in order to implement an edge sensitive interrupt.

A nearby computer is equipped with an OpenMote, a popular IEEE 802.15.4-based platform [67]. The OpenMote is programmed to listen to a particular frequency, and report to the computer the frames it receives. Upon measuring the timings from the laser pulses, the SCμM chip reports those values to the computer over 2.4 GHz IEEE 802.15.4. The Python-based program running on the computer then uses the timings to calculate the relative azimuth and elevation angles between each lighthouse base station and the SCμM chip. Chapter 7 provides extensive detail on the implementation of this code.

4.4 Single Lighthouse Validation Experiments [3]

Experimental Setup

An experiment was designed to test the tracking performance of the optical receiver when using a commercial-off-the-shelf HTC Vive V1 lighthouse. The optical receiver chip was moved through a trajectory while tracking itself using the IR pulses and sweeps from the lighthouse beacon. The ground truth of the receiver's trajectory was obtained using an Optitrack motion capture system. This system was synchronized to the sync pulse of the lighthouse beacon to avoid infrared interference. The experimental setup is illustrated in Figure 4.7, which shows the ground truth trajectory of the receiver in relation to the position and orientation of the lighthouse beacon. Only one lighthouse beacon was used, so the optical receiver system was calculating its azimuth and elevation relative to the lighthouse beacon, not its full 3D location which would require two lighthouse beacons to calculate. The azimuth and elevation of the receiver in this experiment are approximately proportional to the y and

[3]adapted from [65]

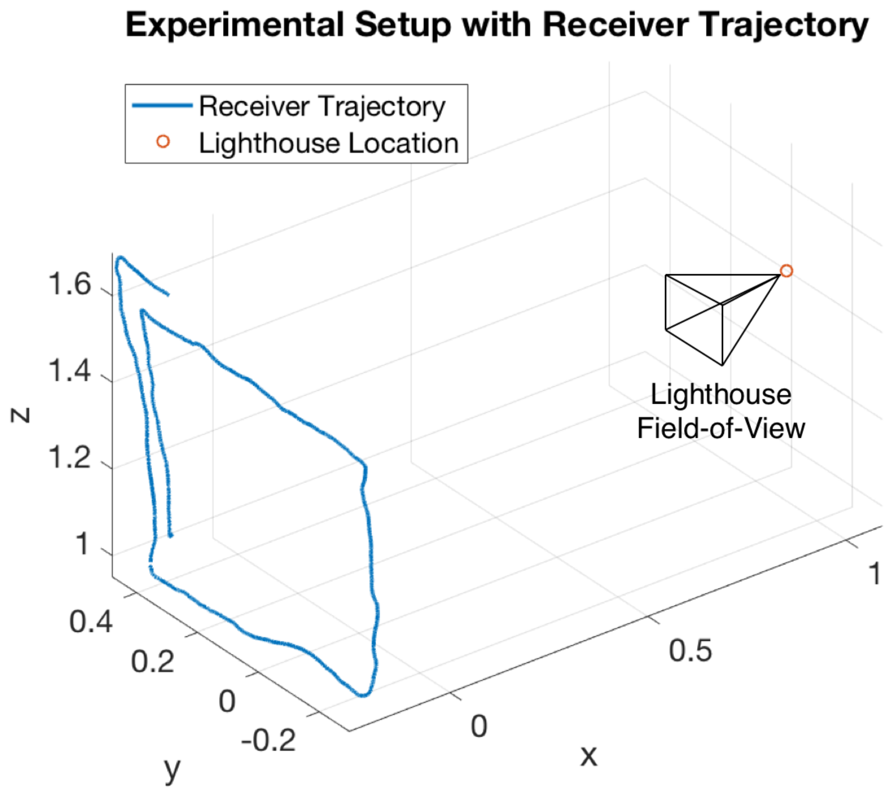


Figure 4.7: Illustration of lighthouse experimental setup. The blue line indicates the ground truth 3D trajectory of the optical receiver, obtained by motion capture. The red dot indicates the location of the lighthouse beacon, and the field-of-view indicates the direction the lighthouse beacon is pointing.

z coordinates of the receiver, respectively. The receiver was approximately 1.3 meters away from the lighthouse throughout the experiment.

Experimental Results

The measured azimuth and angle trajectory are compared in Figures 4.8 and 4.9. Note that the azimuth and elevation of the receiver is approximately proportional to its trajectory projected onto the $(-y, z)$ plane. A post-hoc calibration to take into account discrepancies in lighthouse position and alignment relative to ground truth was applied to the data. This consisted of a lighthouse azimuth alignment adjustment of 2.5 degrees, a lighthouse elevation alignment adjustment of 0.5 degrees, and a lighthouse centroid adjustment of 5 cm. These numbers were produced by manually iterating to minimize error. A simple filtering algorithm was used to reject outlying measurements. This algorithm rejected measurements that were

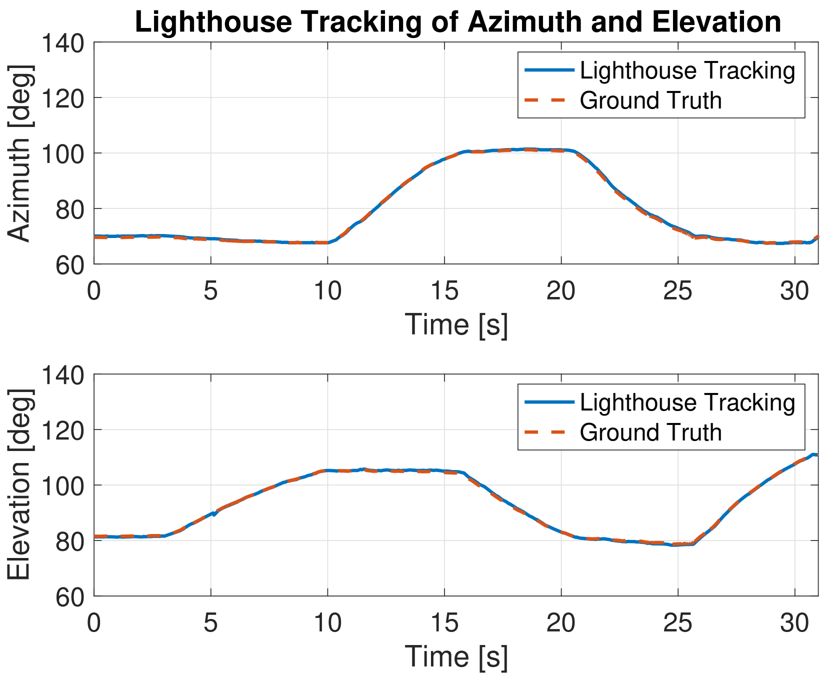


Figure 4.8: Azimuth and elevation tracking of optical receiver, compared to ground truth. The RMS error of azimuth tracking was 0.386 degrees. Elevation tracking's RMS error was 0.312 degrees.

more than 5 degrees different than the previous measurement. The RMS error of the azimuth measurements was 0.386 degrees, which would correspond to a 9 mm error at a distance of 1.3 m. The RMS error of the elevation measurements was 0.312 degrees which corresponds to a 7 mm error at a distance of 1.3 m.

4.5 Lighthouse Triangulation

Lighthouse Projection Model

Our work in [65] demonstrates the ability of a single lighthouse base station to provide accurate relative angle measurements between itself and a SCμM mote. A single base station, however, doesn't provide enough information for 3D localization of a single, freely moving mote; relative angle measurements from another base station with known relative pose (translation and rotation) from the original base station are required to triangulate 3D position. Additionally, triangulation in three dimensions proves a challenge since the two directional rays, generated by the relative angle measurements from each lighthouse, are unlikely to intersect in 3D space due to inherent noise in the measurements. As a result, an estimation method is required to find the triangulation solution that minimizes the error

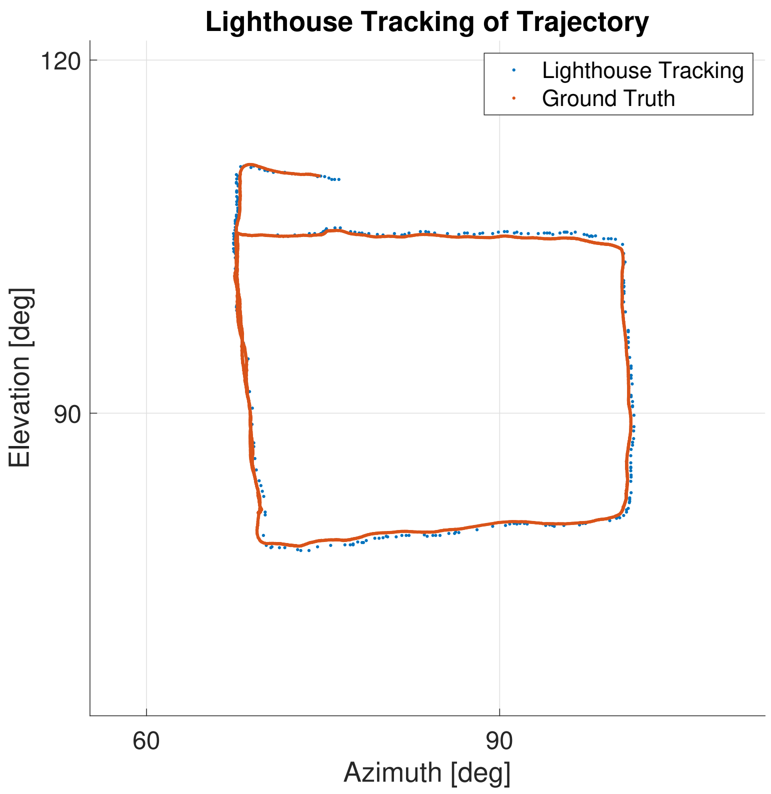


Figure 4.9: Performance of azimuth and elevation tracked by the optical receiver compared to the ground truth trajectory.

between the two rays (see Fig. 4.2).

Triangulation is a well-studied problem commonly arising in computer vision with multi-perspective cameras [68]. In fact, modeling each lighthouse base station as a camera allows for both calibration of relative poses of each lighthouse in addition to the use of triangulation methods like the Direct Linear Transform (DLT) for 3D localization [68, 69]. The mathematical model that describes the transformation between a point in 3D global coordinates to a 2D point on the image plane of a pinhole camera is described by (4.1).

$$X_{image} = \begin{bmatrix} x^{img} \\ y^{img} \\ 1 \end{bmatrix} = PX_{global} \tag{4.1}$$

Eq. (4.2) is the definition of the camera projection matrix, which projects the 3D object onto the 2D camera's image plane. K is the matrix representing the intrinsic camera parameters, which are the focal length f_x, f_y (the distance between the focal plane and the pin hole) and the principal point offset (c_x, c_y), which is the offset between the center of the image plane and the pinhole. K is expressed in (4.3).

$$P = K[R|t] \tag{4.2}$$

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

The pose matrix $[R|t]$, expressed in (4.4), represents the transformation to rotate and translate points from the global frame to the camera frame. The rotation from global to camera frame is represented by R, which is composed of r_{ij}. The translation from the global frame to the camera frame, in camera frame coordinates, is represented by t_i.

$$[R|t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{4.4}$$

The global point being projected onto the camera's image plane X_{global} is expressed in (4.5). It contains the global point in augmented homogeneous representation [68].

$$X_{global} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.5}$$

In our triangulation implementation, we model the lighthouse base stations as a pinhole camera, where the center of rotation of the lighthouse laser scans is the pinhole and the intrinsic matrix is I. The azimuth and elevation measurements (range 0-π, with $\frac{\pi}{2}$ corresponding to the laser perpendicular to the image plane) are projected at unit distance onto this plane using (4.6). Fig. 4.10 illustrates this model.

$$X_{image} = \begin{bmatrix} x^{img} \\ y^{img} \\ 1 \end{bmatrix} = \begin{bmatrix} \tan(\theta_{azimuth} - \frac{\pi}{2}) \\ \tan(\theta_{elevation} - \frac{\pi}{2}) \\ 1 \end{bmatrix} \tag{4.6}$$

Triangulation

We use the Direct Linear Transform (DLT) [68] to triangulate the position of the chip using the relative angle measurements from each lighthouse base station. In the DLT, a system of equations, derived from (4.7) is set up using the camera projection matrices of each base station from the unknown global point. This system is shown in (4.8), where $\mathbf{p}^j$ is the j^{th} column of each base station's camera projection matrix. The projection matrix and image coordinates of the second lighthouse base station are denoted P' and x', y'. This system of equations is solved using least squares[4]. The most recent of each azimuth and

[4] available at `https://github.com/PisterLab/scum_lighthouse_localization`.

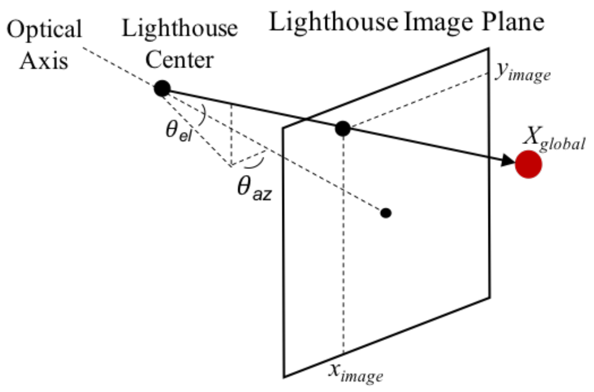


Figure 4.10: Camera model of a lighthouse base station. The azimuth and elevation measurements are projected onto the virtual unit distance image plane of the lighthouse.

elevation measurements are used in the triangulation algorithm. Unfortunately this can lead to inaccuracies if one of these measurements are missed as an out-of-date measurement is then used in its place.

$$PX = \begin{bmatrix} x_{image} & y_{image} & 1 \end{bmatrix}^T \tag{4.7}$$

$$A = \begin{bmatrix} x\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} - \mathbf{p}^{2T} \\ x'\mathbf{p}'^{3T} - \mathbf{p}'^{1T} \\ y'\mathbf{p}'^{3T} = \mathbf{p}'^{2T} \end{bmatrix} \tag{4.8}$$

Calibration of Lighthouse Projection Matrices

Using the Direct Linear Transform (DLT) to triangulate the 3D position of SCμM in the lighthouse-camera model requires knowing both the *intrinsic* and *extrinsic* parameters of the two lighthouse base stations, corresponding to the matrices K and $[R|t]$, respectively. Since we know K to be the identity matrix I, the calibration task amounts to estimating the extrinsic parameters for each lighthouse in the lighthouse-camera model. In computer vision, this is known as the Perspective-n-Point problem (or just PnP) [70], whose aim is to determine a camera's pose (position and orientation) given its intrinsic parameters and a set of $n \geq 3$ correspondences between global 3D points and their 2D projections onto the image plane [71].

For the purpose of this experiment, we build a sizeable set of 3D-2D correspondences: we isolate approx. 1000 sample points of the SCμM angle data projected onto the unit distance image plane, and their corresponding ground truth 3D position data. We use the nonlinear Levenberg-Marquardt iterative optimization algorithm to estimate a solution to the PnP problem [71]. The Levenberg-Marquardt algorithm attempts to find a solution $[\hat{R}|\hat{t}]$

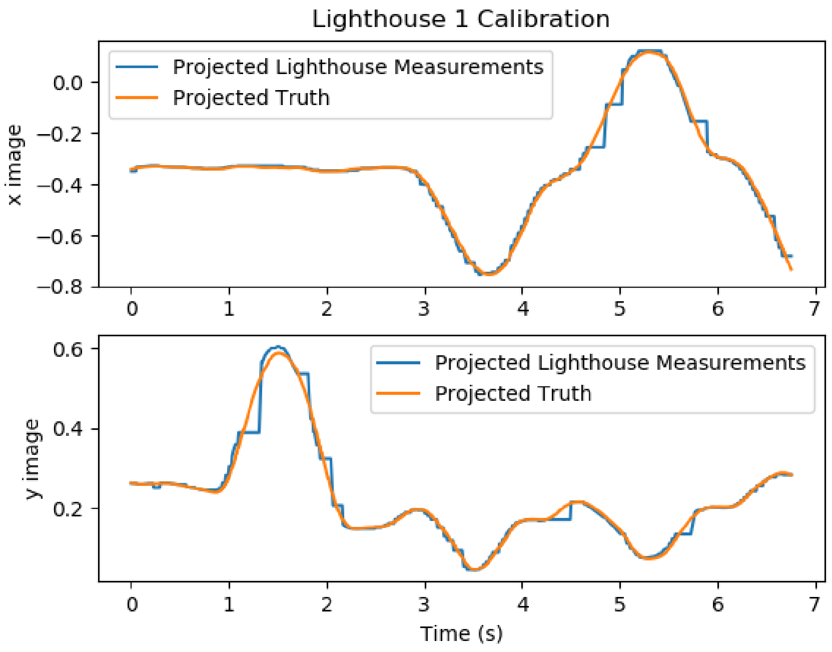


Figure 4.11: Illustration of lighthouse calibration. The calibration process uses the projected measurements and a small portion of the OptiTrack truth trajectory data (this data was not used to calculate measurement accuracy). The PnP algorithm, using the iterative nonlinear least squares Levenberg-Marquardt method, generates the optimal camera projection matrix given this input data. This plot compares the projected lighthouse image points to the ground truth points projected to the lighthouse image plane using the calibrated projection matrix for Lighthouse 1.

to (4.9) by minimizing the residual sum of squared distances between the observed projections $[x_i^{img}\ y_i^{img}\ 1]^T$ and the projected ground truth points $[\hat{R}|\hat{t}][x\ y\ z\ 1]^T$.

$$\begin{bmatrix} x_1^{img} & x_2^{img} & \dots & x_n^{img} \\ y_1^{img} & y_2^{img} & \dots & y_n^{img} \\ 1 & 1 & \dots & 1 \end{bmatrix} = [\hat{R}|\hat{t}] \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ z_1 & z_2 & \dots & z_n \\ 1 & 1 & \dots & 1 \end{bmatrix} \tag{4.9}$$

To implement our calibration procedure, we used OpenCV's `solvePnPRansac` method with the `CV_ITERATIVE` flag [72]. This uses the random sample consensus (RANSAC) iterative method on top of the standard OpenCV Levenberg-Marquardt optimization PnP solver to make the projection robust to outliers in SCμM's angle measurements. We apply the calibration to each lighthouse to recover their relative poses. Fig. 4.11 illustrates the results of using our calibration procedure by applying the pose of Lighthouse 1 that minimizes the reprojection error.

4.6 Results

We evaluate the accuracy of the SCμM lighthouse tracking system using the OptiTrack motion capture system. We move the SCμM chip along a trajectory, while tracking it with both the lighthouse system and the OptiTrack (we use the same methodology as in [65]). We make sure to synchronize the OptiTrack's infrared exposure pulses with the lighthouse sync pulses to avoid interference.

Fig. 4.12 shows the azimuth and elevation angles measured during this experiment by each lighthouse, compared to the ground truth angles. Fig. 4.13 shows the error of the azimuth and elevation tracking; the interquartile range of the measurements is under 1 degree for three of the four angles. The RMS tracking error statistics are skewed by the presence of a few outlier errors. These outliers are observed to be significantly more frequent when the OptiTrack system is active, indicating that the OptiTrack's infrared cameras are likely interfering with the lighthouse signal despite the synchronization protocol. When the outliers are removed in post-processing (error $>$10 degrees), the RMS tracking error is 0.63 deg, 0.37 deg, 0.60 deg, and 3.87 deg. Fig. 4.14 shows the 3D triangulated tracking data for the experiment, compared to ground truth. The mean absolute error for 3D tracking is 1.54 cm, 1.50 cm, and 5.1 cm for the X, Y, and Z dimensions.

We characterize the operating range of the SCμM lighthouse over distance from lighthouse and incidence angle of the SCμM mote relative to the lighthouse pulses. Specifically, we measure the percent of lighthouse measurements received at various distances from a lighthouse base station (Fig. 4.15), with the SCμM mote directly in front of the base station. The maximum effective range of the localization system is 1 m. We measure the effect of the angle of incidence on reliability by rotating the SCμM mote in a stationary position directly in front of two adjacent lighthouse base stations (Fig. 4.16). The maximum angle the SCμM mote could be rotated with respect to the lighthouses is 70°. As SCμM's optical receiver is designed for optical programming and not lighthouse localization, a future redesign of the receiver system could conceivably increase the detection range of the lighthouse signal.

4.7 Localization Improvements

While these results are promising, several problems limit this system's utility. First, intermittent outliers with significant error ($>$10 degrees) are present. Second, this system relies on line-of-sight and is vulnerable to occlusion of the lighthouse base stations. Third, the diminished lighthouse reception rate over distance and incidence angle can intermittently reduce the triangulation rate significantly. Finally, out-of-date measurements can cause inaccuracies in the triangulation algorithm, which relies on four measurements that do not occur simultaneously.

These problems could be solved by fusing the lighthouse measurements with inertial measurement unit (IMU) data. Sensor fusion algorithms like the Extended Kalman Filter (EKF) could use statistical measures such as the Mahalanobis distance to reject outlier

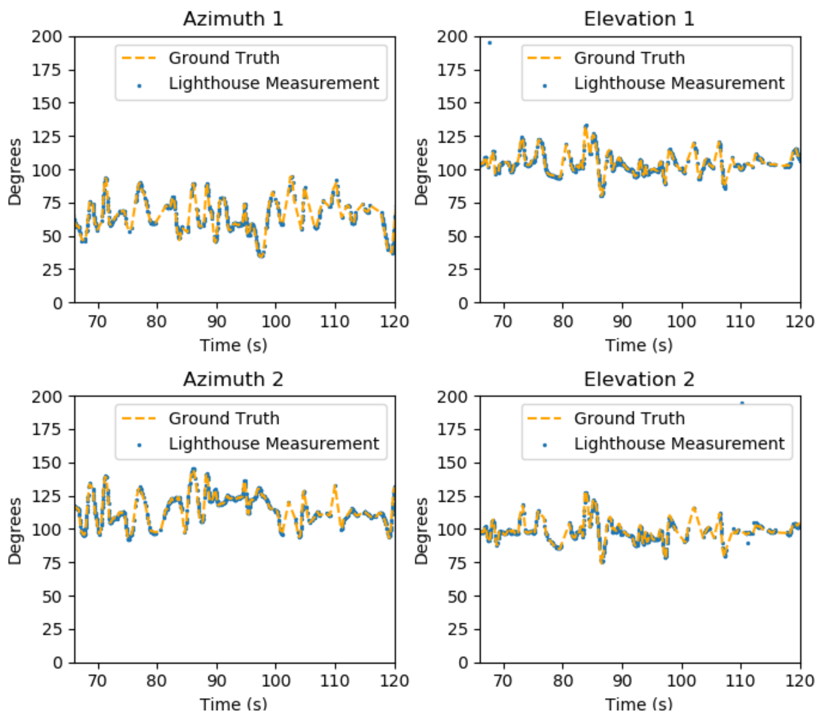


Figure 4.12: Experimental azimuth and elevation measurements of a SCμM mote compared to motion capture ground truth. The lighthouse-relative ground truth azimuth and elevation angles were determined by using the calibrated lighthouse poses to project the 3-dimensional world-frame ground truth trajectory to 2-dimensional lighthouse-relative azimuth and elevation angles.

lighthouse measurements [73, 74]. Furthermore, during lighthouse occlusion events, this state estimator could still calculate the position of the SCμM chip by integrating IMU measurements, improving occlusion tolerance. In fact, we demonstrated in [75] the ability of an EKF to provide occlusion tolerance in a lighthouse-IMU sensor fusion system. Sensor fusion would also increase the tracking rate of the system to the IMU's update rate, which could alleviate the aforementioned lighthouse update rate issues. Finally, an EKF, being a non-linear estimator, could calculate position from direct individual lighthouse angle measurements instead of triangulation from four temporally-disparate measurements.

Currently, the triangulation algorithm is calculated on a computer that is receiving SCμM packets with lighthouse data. This is not scalable to large numbers of SCμM chips, so it would be beneficial to implement the triangulation on-board the SCμM chip itself. Further work must be done to determine the computational load of this triangulation method.

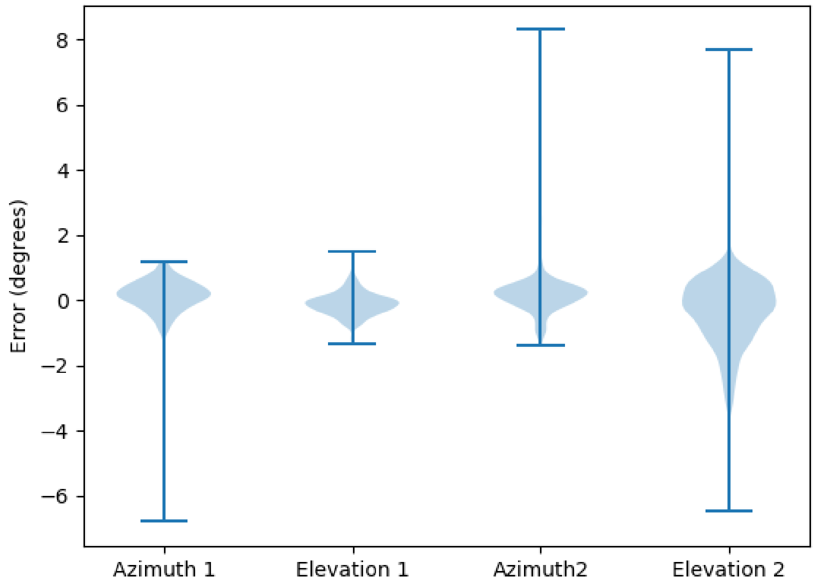
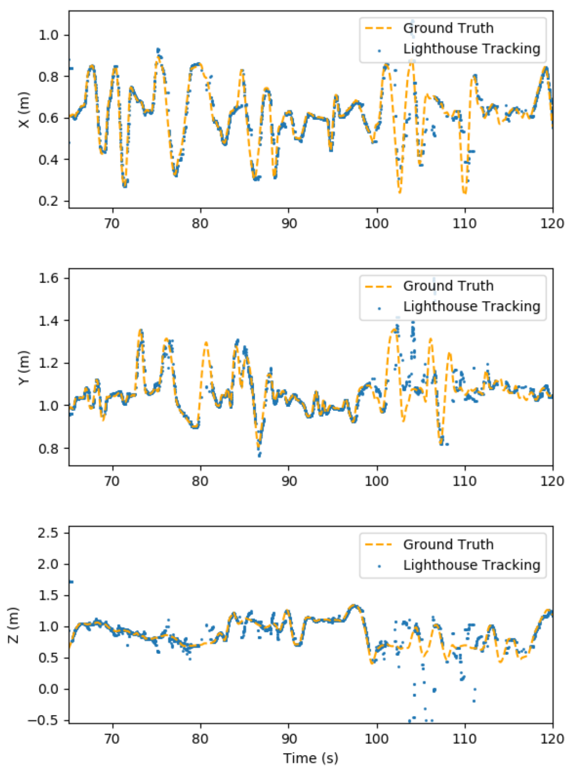


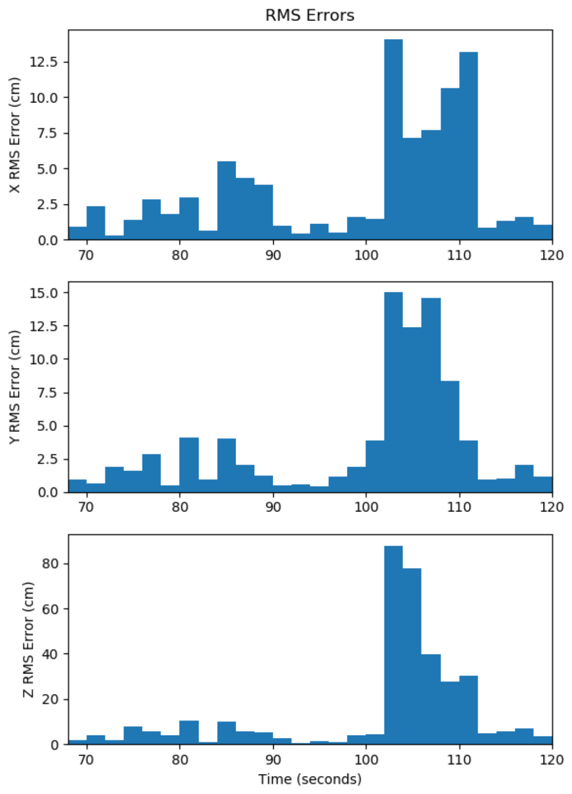
Figure 4.13: Violin plot showing distribution of azimuth and elevation angle error. One outlying data point that is 80 degrees from the mean of the elevation B data is not shown. Data points outside of these percentiles are not shown. The majority of error points are within a tight distribution, with a few outliers far from the mean. The RMS tracking error, with outliers (error >10 degrees) removed, is 0.63 deg, 0.37 deg, 0.60 deg, and 3.87 deg for azimuth 1, elevation 1, azimuth 2, and elevation 2, respectively.

4.8 Conclusion

We have demonstrated the 3D localization of a 1.8 mm^3 crystal-free wireless SoC using SCμM's integrated optical receiver and commercially available virtual reality tracking hardware. This system does not require any extra components on the SCμM side, thus preserving its miniature form factor. Applications in microrobotics and personal telemetry could be improved by this localization system.



(a)



(b)

Figure 4.14: (a) Triangulation of lighthouse data to reconstruct the trajectory of the SCμM chip throughout the experiment. Large deviations in tracking are caused by missed measurements. In our system, a missed measurement gets replaced by the latest measurement. This allows for the system to always display a position, but causes the reconstruction algorithm to use out-of-date measurements, resulting in inaccurate positioning. The subset of data used for lighthouse calibration is excluded from this data. (b) Triangulation error of lighthouse tracking over time. The RMS tracking error has significant temporal deviation throughout the experiment, characterized by periods of time with minimal error and periods of time with significant error. Periods of significant error are likely correlated with line-of-sight occlusion, infrared interference, or RF interference events.

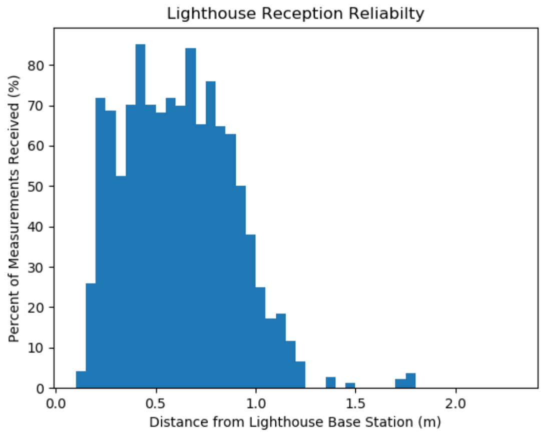


Figure 4.15: Percent of lighthouse measurements received, as a function of distance to lighthouse. The reliability of the lighthouse measurements drops off at around 1 m. An active motion capture system is used in this experiment to measure SCμM position, which could have a small effect on the lighthouse reliability due to infrared interference.

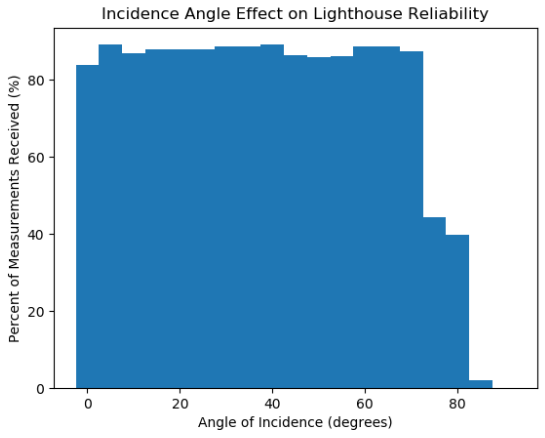


Figure 4.16: Effect of incidence angle on reliability of lighthouse measurements. An angle of 0° corresponds to the face of the SCμM parallel to the face of the lighthouse base station, with the laser scans perpendicular to the face of SCμM. A rapid dropoff in lighthouse measurement reliability occurs when SCμM is rotated $>70^\circ$ relative to the lighthouses. In this experiment, the lighthouses are 9 cm apart, the SCμM was 47 cm away, centered in between the two lighthouses. The incidence angles are measured manually, rather than by a motion capture system.

Chapter 5

SCμM Lighthouse and IMU Sensor Fusion

5.1 Sensor Fusion Advantages

As previously mentioned, there are several weaknesses in the lighthouse localization framework for SCμM that can be mitigated by fusing IMU sensor data with the lighthouse data. The lighthouse localization method for SCμM is sensitive to occlusion, outlying lighthouse measurements, reduced lighthouse localization rate due to reliability decreases, and asynchronous angle measurements. Using sensor fusion to combine IMU measurements with lighthouse localization measurements can mitigate these weaknesses by providing occlusion tolerance, improved nonlinear measurements, localization rate update rate improvement, and statistical outlier rejection. Figure 5.1 illustrates a few of these advantages. For example, if a lighthouse base station is occluded, the SCμM can still estimate its velocity and position, for a short period of time, using sensor fusion, thus providing occlusion tolerance. Furthermore, sensor fusion algorithms can incorporate the individual angle measurements, which are nonlinearly related to position, directly into the their state estimate. Conversely, triangulation requires four angle measurements, so a delayed measurement or lack of measurements from one of the lighthouses can greatly affect the triangulation performance. IMU sensor fusion algorithms can also run at the update rate of the IMU, which usually is higher than the measurement rate of lighthouse localization. Finally, sensor fusion can use statistical measures, such as the Mahalanobis distance, to detect outliers [73] (Figure 5.2). Equation 5.17 shows the definition of the Mahalanobis distance, and is further discussed later in this chapter.

5.2 IMU and Lighthouse Implementation on SCμM

Interfacing an IMU with SCμM is required to enable sensor fusion. The ICM20948, manufactured by TDK/Invensense, is a suitable IMU since it has a small 3 x 3 x 1 mm^3 package and can run on a voltage supply of 1.8V, which is compatible with SCμM. The ICM20948

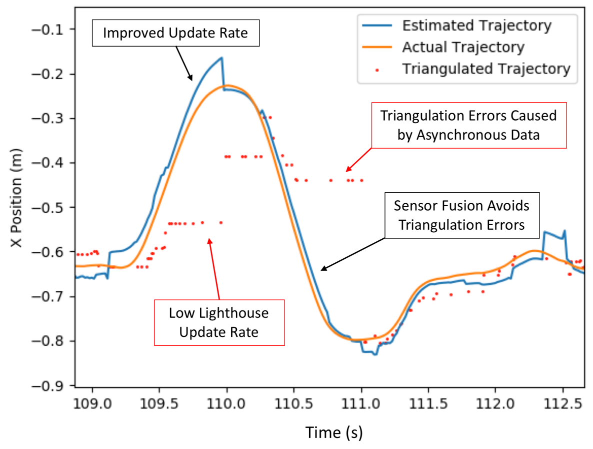


Figure 5.1: Selected advantages of IMU sensor fusion for improving lighthouse localization on SCμM. Sensor fusion tracking systems can update at the same rate as the IMU and can provide a tracking estimate even when lighthouse measurements are occluded or aren't available. Nonlinear estimation enabled by sensor fusion can also avoid issues that arise when triangulating using multiple asynchronous measurements that may be delayed. Nonlinear estimation allows the integration of individual lighthouse angle measurements, rather than requiring four asynchronous angle measurements to generate a triangulated measurement.

contains a 3-axis accelerometer, a 3-axis gyroscope, and a 3-axis magenetometer. SCμM interfaces with the IMU via an SPI bus. As there is no dedicated SPI module on SCμM, the SPI interface is implemented by bit-banging GPIOs[1].

The IMU data is read via a polling scheme, which is feasible since the lighthouse measurement processing is interrupt-based. The processor continuously polls the IMU for data, and processes lighthouse measurements when relevant interrupts are triggered. It should be noted that lighthouse measurement interrupts likely cause communication errors between the IMU and the processor if an interrupt occurs in the middle of an SPI read or write operation. The code below shows the implementation of this polling loop:

```
//start localization loop
while(1) {

   //declare measurement struct
   imu_data_t imu_measurement;


   //get imu measurements
   imu_measurement.acc_x.value = read_acc_x();
```

[1]code available at https://github.com/PisterLab/scum-test-code

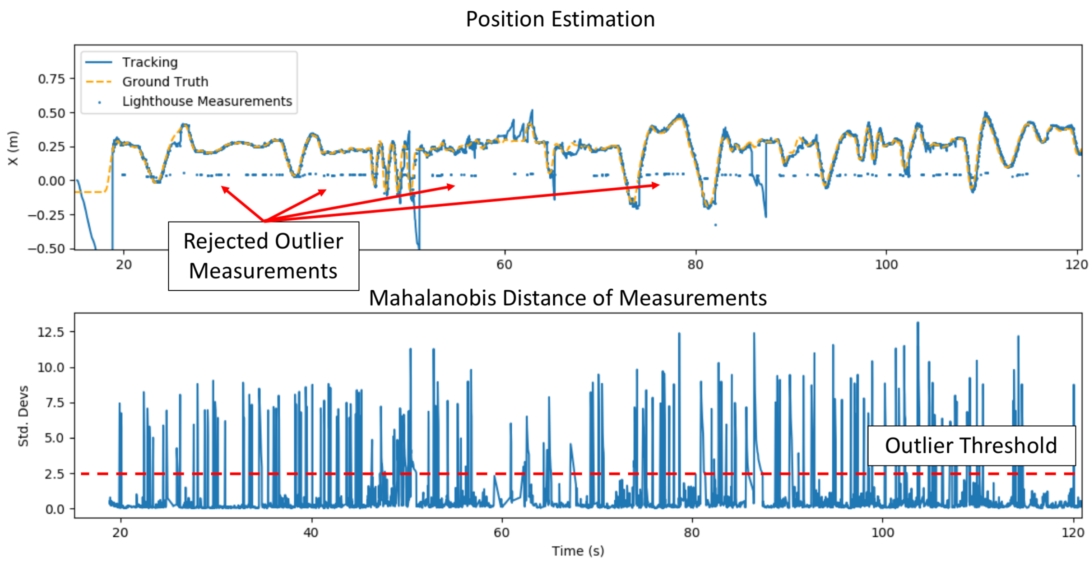


Figure 5.2: Statistical outlier rejection using Mahalanobis distance [76]. This estimator is using SCμM lighthouse data with IMU data. State estimation techniques used in sensor fusion keep a continuous estimate the probability distribution of the SCμM's position and state. Using this probability distribution and a model providing predicted lighthouse measurements as a function of SCμM state, sensor fusion algorithms can determine the likelihood of a measurement being an outlier. The Mahalanobis distance is one measure of this, which is roughly the number of standard deviations a measurement is from its predicted measurement, given SCμM's current state.

```
imu_measurement.acc_y.value = read_acc_y();
imu_measurement.acc_z.value = read_acc_z();
imu_measurement.gyro_x.value = read_gyro_x();
imu_measurement.gyro_y.value = read_gyro_y();
imu_measurement.gyro_z.value = read_gyro_z();

//wait; this determines imu data rate
//this should really be replaced by a timer trigger

for( i = 0; i < 10000; i++){
}

//send measurement
send_imu_packet(imu_measurement);
```

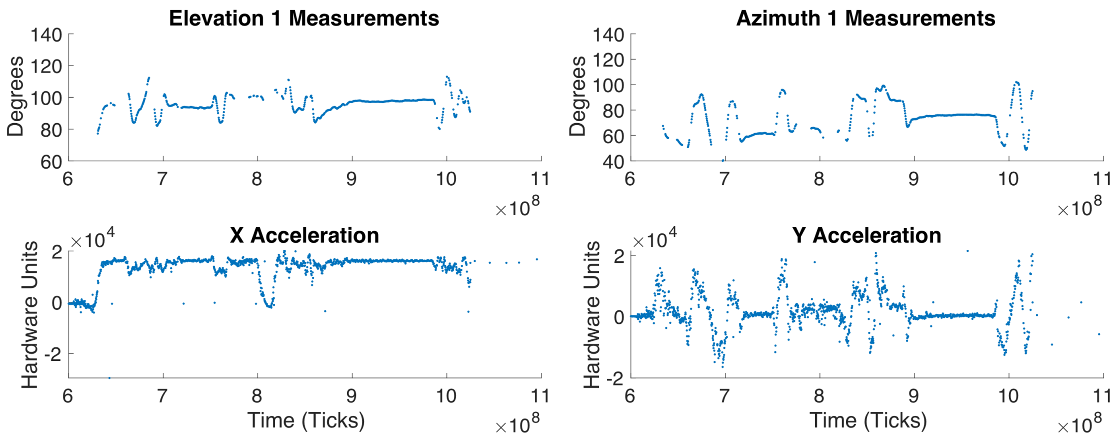


Figure 5.3: Lighthouse and IMU data gathered by SCμM during a brief experiment. At constant z position, azimuth measurements are proportional to x position and elevation measurements are proportional to y position, given that $\frac{x}{z} = tan(\theta_{az})$ and $\frac{y}{z} = tan(\theta_{el})$. As a result, the second derivative of these angles is roughly equivalent to the IMU accelerations measured on those axes. The IMU data rate was 50 Hz in this experiment. The x axis is 10 Mhz timer ticks, so each tick corresponds to 100 μs.

```
}
```

At the end of each loop iteration, SCμM will transmit a packet containing IMU data. The amount of wait time in the polling loop determines the rate at which the IMU is polled and packets are sent. Observations indicate that this loop can be set to run at up to 120 Hz before lighthouse pulse processing performance is significantly affected. The maximum IMU data rate observed, without lighthouse processing running simultaneously, is approximately 200 Hz. Figure 5.3 shows experimental data of SCμM processing IMU and lighthouse data as it travels through a trajectory in space.

5.3 State Estimation and Sensor Fusion Algorithms

A common algorithm used for implementing sensor fusion is the Kalman Filter [77], which is a recursive state estimator that incorporates system dynamics in order to determine the state of a system based on noisy measurements. The Kalman Filter maintains an estimate of the probability distribution of a system's noisy states. It contains two steps: a propagation step where the current state estimate is propagated through the system's dynamic model, and an update step where a measurement of the system's state is incorporated into the state estimate (Figure 5.4). The Kalman Filter is the optimal bayesian state estimator (minimizing mean squared error) for linear systems with gaussian zero-mean noise and disturbances. While the Kalman filter is not an optimal estimator for nonlinear systems or nongaussian noise, it is still the best *linear* state estimator for these cases. Additional state estimators and variations of the Kalman filter have been developed for nonlinear and non-gaussian systems such as the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) [78], and the particle filter [79].

Linear Kalman Filter Algorithm

The linear Kalman filter can be used to estimate the state of a linear system from noisy observations, assuming that system disturbances and measurement noise are zero-mean and gaussian. Consider the following linear system that is observed through measurements:

$$\vec{x}(k) = A(k-1)\vec{x}(k-1) + u(k-1) + v(k-1) \tag{5.1}$$

$\vec{x}(k)$: state $A(k)$: system model $u(k)$: system input $v(k)$: process noise

The measurement model for a measurement observing this system is defined below:

$$z(k) = H(k)\vec{x}(k) + w(k) \tag{5.2}$$

$z(k)$: observed measurement $H(k)$: linear measurement model $w(k)$: measurement noise

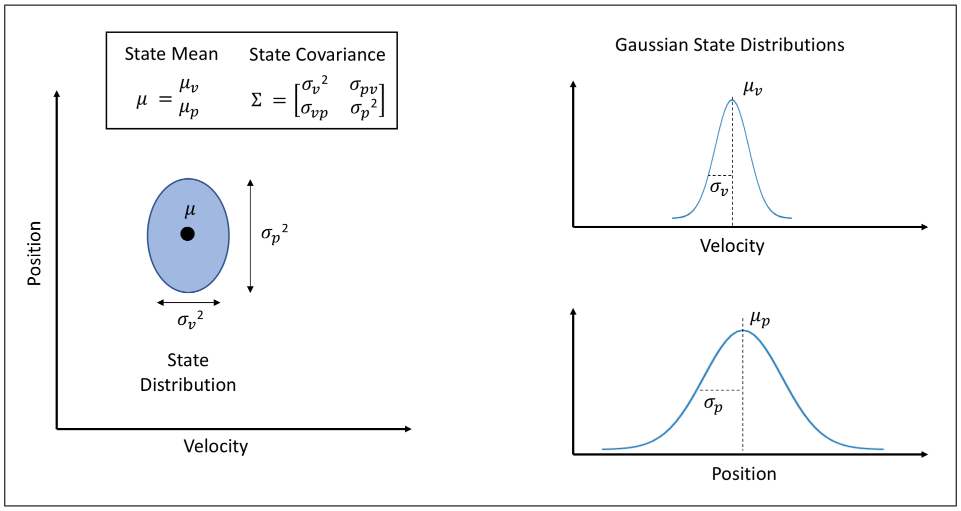
The linear Kalman filter estimates the state, $\vec{x}$, based on system inputs and observed measurements of the system. It performs this using two steps: a prediction step and an update step. The prediction step generates a prior state estimate, $\hat{x}_p(k)$, from the posterior state estimate from the previous timestep, $\hat{x}_m(k-1)$, the input, $u(k-1)$, and the input model, B. A prior state covariance for the current timestep, $P_p(k)$, is also created during the prediction step using the covariance of $v(k)$, which is denoted by Q. The distribution of the process noise is assumed to be zero mean and gaussian. Generally in inertial guidance systems, the IMU measurements are incorporated into the prediction phase of the algorithm, usually as inputs.

Prediction Step

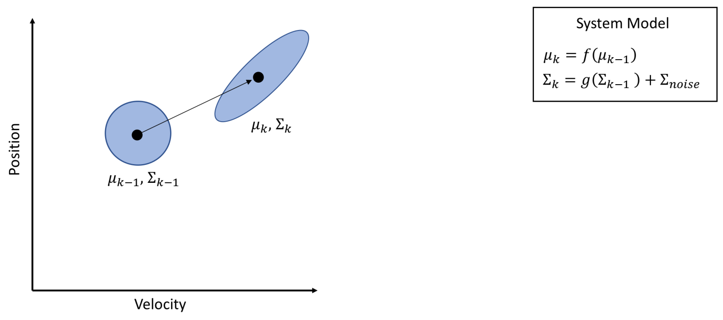
$$\hat{x}_p(k) = A(k-1)\hat{x}_m(k-1) + Bu(k-1) \tag{5.3}$$

$$P_p(k) = AP_m(k-1)A^T + Q \tag{5.4}$$

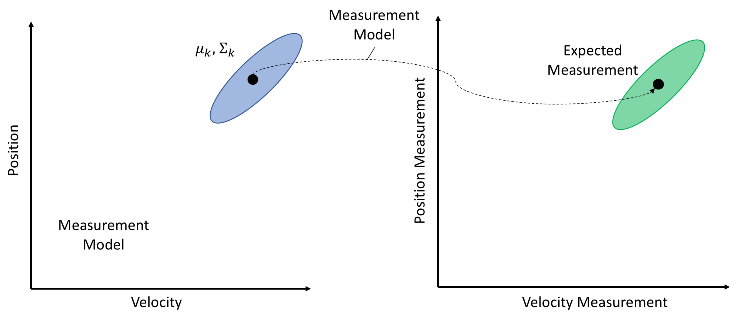
After the prediction step occurs and a measurement observation is available, the measurement update step can be performed. During this step, a predicted measurement covariance, or "innovation" covariance (denoted as S), is calculated (Equation 5.5). In this calculation, R is the covariance of the measurement noise, $w(k)$, and is assumed to be zero mean and gaussian. The kalman gain, $K(k)$, is then calculated (Equation 5.6). The posterior estimate, $\hat{x}_m(k)$, is then calculated based on the Kalman gain and the difference between the observed


State Mean
State Covariance
$\mu = \begin{matrix}\mu_v \\ \mu_p\end{matrix}$
$\Sigma = \begin{bmatrix}\sigma_v{}^2 & \sigma_{pv} \\ \sigma_{vp} & \sigma_p{}^2\end{bmatrix}$
Position
μ
$\sigma_p{}^2$
$\sigma_v{}^2$
State Distribution
Velocity
Gaussian State Distributions
μ_v
σ_v
Velocity
μ_p
σ_p
Position

(a) State distribution model

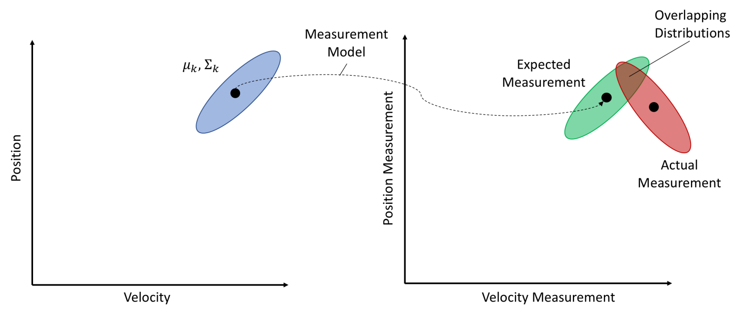

System Model
$\mu_k = f(\mu_{k-1})$
$\Sigma_k = g(\Sigma_{k-1}) + \Sigma_{noise}$
Position
μ_k, Σ_k
μ_{k-1}, Σ_{k-1}
Velocity

(b) State propagation step. A prior for the current state of the model is estimated by propagating the previous posterior state estimate through the dynamics model.

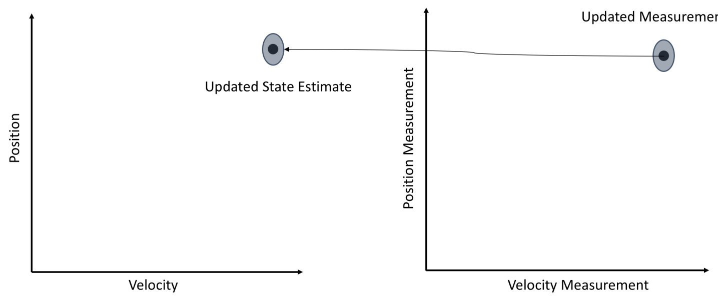


(c) The current measurement is predicted by propagating the current state estimate prior through a measurement model.

Figure 5.4: Illustration of Bayesian state estimation



(d) Measurement step: when a measurement of the current state is acquired, a distribution of the measurement is estimated using the noise distribution of the measurement.



(e) Measurement Update: Bayes rule is used to obtain the current posterior state estimate, given the prior state and state measurement.

Figure 5.4: Illustration of Bayesian state estimation

measurement, z, and the expected measurement given the prior state, $H(k)\hat{x}_p(k)$ (Equation 5.7). The posterior covariance estimate, $P_m(k)$ is then updated (Equation 5.8).

Measurement Update Step

$$S = H(k)P_p(k)H(k)^T + R \tag{5.5}$$

$$K(k) = P_p(k)H^T(k)S^{-1} \tag{5.6}$$

$$\hat{x}_m(k) = \hat{x}_p(k) + K(k)(z(k) - H(k)\hat{x}_p(k)) \tag{5.7}$$

$$P_m(k) = (I - K(k)H(k))P_p(k) \tag{5.8}$$

Nonlinear extensions to this filter, such as the EKF and UKF, still follow the predict and update step algorithm prototype that is used in the linear Kalman filter.

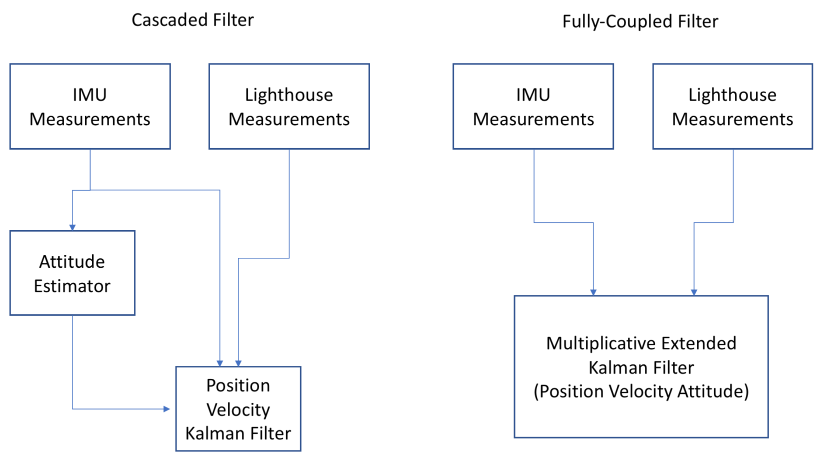


Figure 5.5: Decoupled architecture vs. fully-coupled architecture.

5.4 State Estimation for SCμM

For localizing SCμM, the state estimation algorithm must estimate SCμM's position, velocity, and attitude. The linear Kalman filter is suitable for estimating the position and velocity, given the triangulated position measurements that lighthouse localization produces. However, since the IMU measurements are in reference to the body frame of the SCμM and not the global frame of reference, the attitude of the SCμM must be known. Unfortunately these systems dynamics are no longer linear, so extensions to the linear Kalman filter must be used. Different architectures could be used to solve this problem. For example, a decoupled filter architecture, which separates the attitude estimation from the linear position and velocity estimation could be used. A fully coupled architecture could also be used, where the attitude estimation is fully integrated with the position and velocity estimation, such as the Multiplicative Extended Kalman Filter (MEKF) [80, 81]. The MEKF has been used to successfully track micro unmanned aerial vehicles [82, 83] and HTC Vive lighthouse systems [84]. Figure 5.5 illustrates the differences between these architectures. One advantage of using a decoupled architecture for SCμM is that it can reduce computational load by leveraging the ICM20948's Digital Motion Processor (DMP) to estimate attitude.

Decoupled Filter Architecture

The proposed cascaded filter architecture uses a Madgwick Filter [85] to estimate attitude and a Kalman filter to estimate the position and velocity. The Madgwick filter could eventually be replaced by the attitude estimation engine onboard the ICM20948's DMP, and the Kalman filter can be implemented as a linear filter or as a nonlinear extension of the Kalman filter. The attitude estimate is used at each step of the filter to transform the body-frame accelerometer measurements to the global frame, where they are used by the position-velocity estimator.

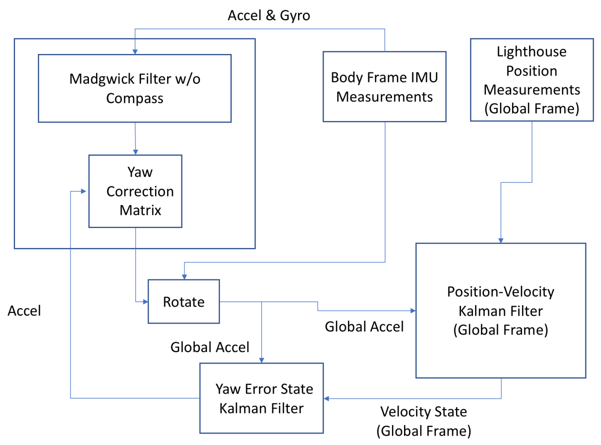

Accel & Gyro
Madgwick Filter w/o Compass
Body Frame IMU Measurements
Lighthouse Position Measurements (Global Frame)
Yaw Correction Matrix
Accel
Rotate
Global Accel
Position-Velocity Kalman Filter (Global Frame)
Global Accel
Yaw Error State Kalman Filter
Velocity State (Global Frame)

Figure 5.6: Proposed cascaded filter architecture for lighthouse sensor fusion.

Figure 5.6 shows the architecture of this proposed filter.

SCμM Dynamics

Equation 5.9 shows the estimator state, where p is the position of the SCμM in the global frame, and v is the velocity of the SCμM in its body frame. Equation 5.10 shows the relationship between the actual measurement, z, and the measurement model, $h(\vec{x})$, which is based on the current state of the SCμM. The measurement model depends on if triangulated measurements are used as input or direct angle measurements are used.

$$\vec{x} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{bmatrix} \tag{5.9}$$

$$z = h(\vec{x}) \tag{5.10}$$

The prediction step incorporates the IMU measurements, acceleration ($\vec{a}$) and gyroscope ($\vec{\omega}$), into the dynamics of SCμM's motion. Equation 5.11 shows the dynamics step used for the Kalman filter, where $\vec{p}$ is position, $\vec{v}$ is velocity, R_g^b is the rotation matrix for the transformation from the body reference frame to the global frame, g is acceleration due to gravity, Δt is the timestep length, and $\vec{e_3}$ is the unit vector $\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$.

$$\begin{aligned} \dot{\vec{v}} &= \vec{a} - \lfloor \vec{\omega}(k) \rfloor_\times v(t) - g {R_g^b}^T \vec{e_3} \\ \vec{p}(k) &= \vec{p}(k-1) + R_g^b \vec{v}(k-1) \\ \vec{v}(k) &= \vec{v}(k-1) + \dot{\vec{v}} \Delta t \end{aligned} \tag{5.11}$$

The system matrix, A, used in the Kalman filter equations is shown in Equation 5.12.

$$A = \begin{bmatrix} I_3 & R_g^b I_3 \Delta t \\ 0_{3x3} & I_3 \end{bmatrix} \tag{5.12}$$

The measurement model, $H(k)$, depends on the implementation of the Kalman filter. The linear Kalman filter requires the triangulated lighthouse measurements, while a nonlinear filter can use direct angle measurements as an update. The next section will further detail these models.

Unscented Kalman Filter

The UKF is a form of the Kalman filter that can handle nonlinear dynamics [78]. It models the statistics of the state and measurements with a select number of "sigma points". These representative sigma points are propagated through the dynamics and measurement models in order to obtain the prior state and measurement estimates. The covariance and means of these sigma points are then calculated and used to obtain the Kalman gain of the filter. The measurement and dynamics models used in the UKF can be nonlinear. When applied to the lighthouse sensor fusion problem, the UKF can use direct lighthouse angle measurements, instead of the triangulated position measurements.

Lighthouse Measurement Model

Two implementations of the position estimator were implemented: one based on a linear Kalman filter and one based on an Unscented Kalman Filter (UKF) [78]. Both filters used the same dynamics model, but different measurement models: the linear Kalman filter used the triangulated lighthouse measurements (Equation 5.13), and the UKF used the direct azimuth and elevation measurements relative to the lighthouses (Equation 5.16). Figure 5.7 illustrates the difference between these two models.

$$z = h(\vec{x}) = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{5.13}$$

$$z = \text{triangulate}(\theta_{az1}(t+\epsilon_1), \theta_{el1}(t+\epsilon_2), \theta_{az2}(t+\epsilon_3), \theta_{el2}(t+\epsilon_4)) \tag{5.14}$$

$$\begin{aligned} z(k) &= H(k)\vec{x} \\ H(k) &= \begin{bmatrix} I_3 & 0_{3x3} \end{bmatrix} \end{aligned} \tag{5.15}$$

In Equation 5.14, $t + \epsilon_n$ denotes that each lighthouse angle measurement is from a slightly different time. If the value of ϵ_n grows too large for any measurement (e.g. one lighthouse is occluded for a short period of time), localization errors will grow. One method of avoiding this is gating angle measurements based on a maximum threshold value of ϵ_n and requiring

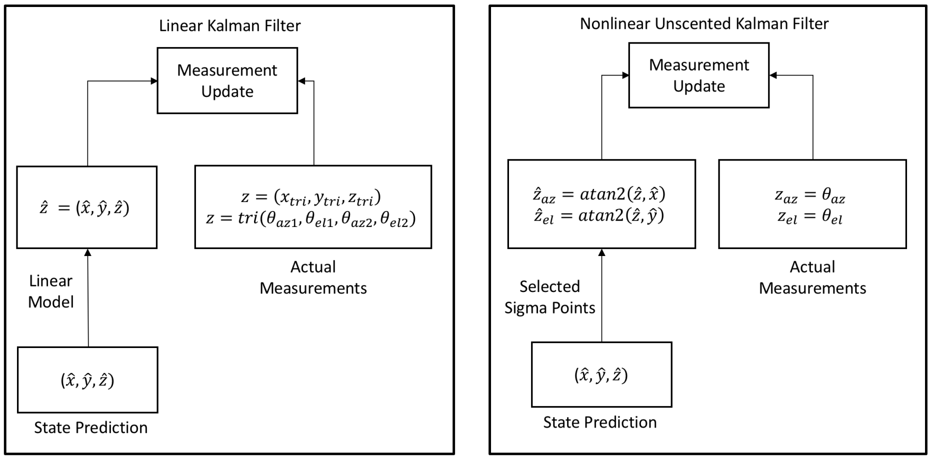

Linear Kalman Filter
Measurement Update
$\hat{z} = (\hat{x}, \hat{y}, \hat{z})$
$z = (x_{tri}, y_{tri}, z_{tri})$
$z = tri(\theta_{az1}, \theta_{el1}, \theta_{az2}, \theta_{el2})$
Actual Measurements
Linear Model
$(\hat{x}, \hat{y}, \hat{z})$
State Prediction
Nonlinear Unscented Kalman Filter
Measurement Update
$\hat{z}_{az} = atan2(\hat{z}, \hat{x})$
$\hat{z}_{el} = atan2(\hat{z}, \hat{y})$
$z_{az} = \theta_{az}$
$z_{el} = \theta_{el}$
Actual Measurements
Selected Sigma Points
$(\hat{x}, \hat{y}, \hat{z})$
State Prediction

Figure 5.7: Comparison of linear kalman filter architecture to UKF architecture.

that all measurements be within a certain period of time. The downside of this approach is that useful data could be thrown away, decreasing the effective localization rate. The azimuth and elevation measurements from lighthouse 1 are denoted θ_{az1} and θ_{el1}, while the measurements from lighthouse 2 are denoted θ_{az2} and θ_{el2}. The measurement model, $H(k)$, that is used in the Kalman filter is given in Equation 5.15.

The UKF implementation avoids this problem and provides a distinct advantage as it only requires a single angle measurement in order to execute the update step of the filter, while the linear Kalman filter requires all four lighthouse angle measurements. In Equation 5.16, the measurement model can be one of four cases: an azimuth measurement from lighthouse 1 or 2, or an elevation measurement from lighthouse 1 or 2. The location of each lighthouse i is represented by x_{li}, y_{li}, z_{li} in Equation 5.16.

$$h(\vec{x}) = \begin{cases} \text{atan2}(z - z_{l1}, x - x_{l1}) & \text{if } z = \theta_{az1} \\ \text{atan2}(z - z_{l1}, y - y_{l1}) & \text{if } z = \theta_{el1} \\ \text{atan2}(z - z_{l2}, x - x_{l2}) & \text{if } z = \theta_{az2} \\ \text{atan2}(z - z_{l2}, y - y_{l2}) & \text{if } z = \theta_{el2} \end{cases} \tag{5.16}$$

5.5 Performance with Ideal Attitude Estimation

These sensor fusion architectures were first validated by using experimental data from a SCμM triangulation experiment. No IMU data was present in this experiment, so the acceleration and angular velocity of the SCμM we derived from the OptiTrack ground truth data. The attitude estimation was also replaced with the ground truth attitude, as to isolate the position-velocity estimator's performance (Fig. 5.8).

The performance of both estimators was evaluated on the experimental data using the derived acceleration and ground truth attitudes. Figure 5.9 shows these results. A significant

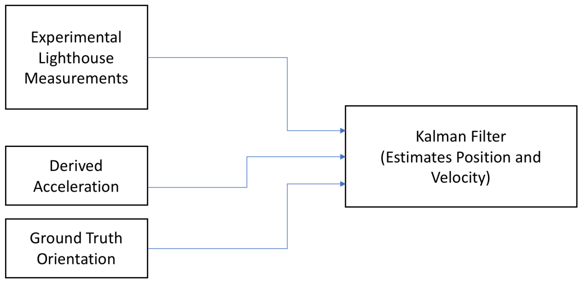

Experimental Lighthouse Measurements
Derived Acceleration
Ground Truth Orientation
Kalman Filter (Estimates Position and Velocity)

Figure 5.8: Architecture used to evaluate position estimation, independent of attitude estimation. Accelerometer and gyroscope measurements were derived from the ground truth pose data from the experiment. Attitude was taken directly from the ground truth data in order to isolate the performance of the position estimator.

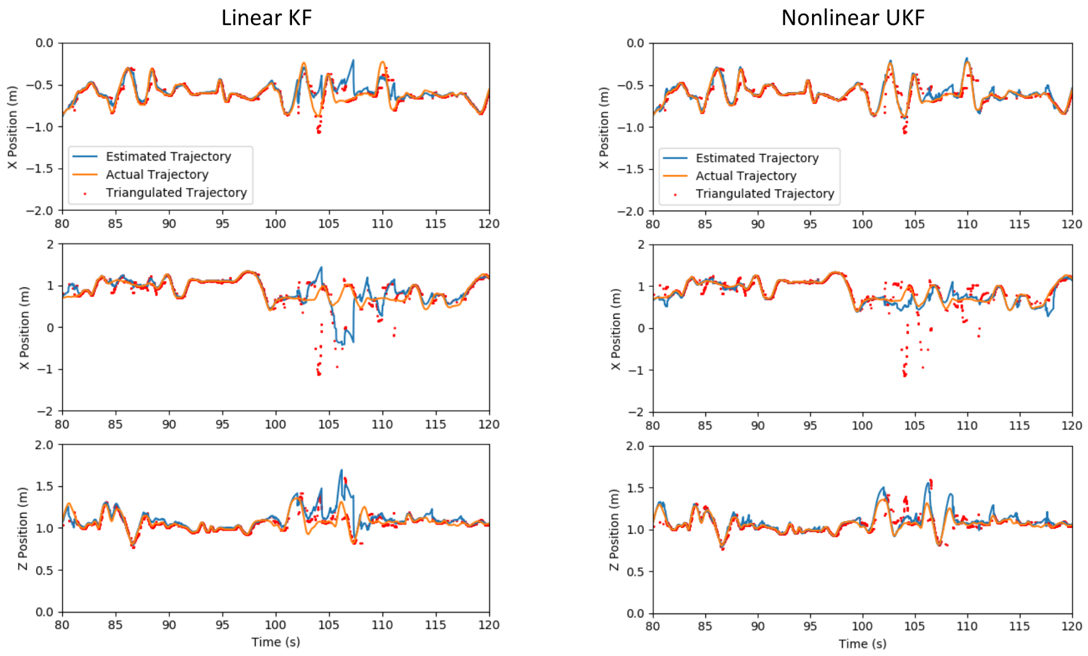


Figure 5.9: Tracking comparison between the linear and nonlinear state estimators. The UKF improves performance in areas where some measurements lose reliability do to occlusion or lost measurements, such as around 105 seconds in the plots above.

improvement can be seen in the UKF implementation in situations where there are delayed or missing angle measurements (Figure 5.10). One caveat with this non-linear filtering is that it is somewhat less robust to tuned parameters such as the process covariance (Figure 5.11). This behavior is not seen with the linear Kalman filter implementation.

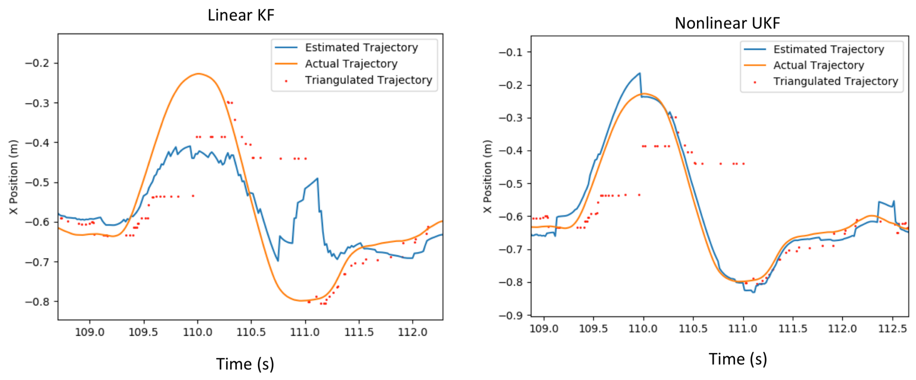


Figure 5.10: Selected example showing the benefits of nonlinear estimation. The linear Kalman filter is limited to using the triangulated lighthouse measurements, which in this case are inaccurate due to outdated angle measurements being used in the triangulation function. There are still some lighthouse measurements being received, but all four of the necessary angle measurements aren't being updated, so the triangulation algorithm uses the most recent measurements. The UKF can handle this error case, since it can use each individual measurement to update, rather than needing four measurements that are asynchronous.

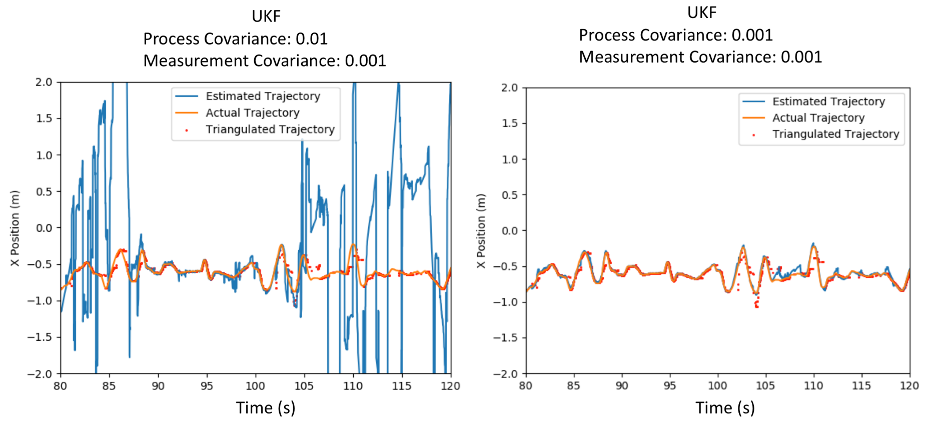


Figure 5.11: Sensitivity of UKF to process covariance. While the UKF provides improvements over the Kalman filter for lighthouse localization, it is more sensitive to calibrated parameters such as the process and measurement covariances, and it can diverge at certain parameters. This can increase the difficulty of UKF implementation compared to the Kalman filter, which doesn't experience this issue.

5.6 Outlier Rejection with Mahalanobis Distance

The Mahalanobis distance [76] is one statistical metric that can be used to reject ouliers in state estimation algorithms [73]. It measures the number of standard deviations a point is from the mean of a distribution. In this case, the Mahalanobis distance is used to determine the distance that a lighthouse measurement is from the expected measurement distribution. Equation 5.17 show the definition of the Mahalanobis distance, where z is the actual lighthouse measurement, $\hat{z}$ is the predicted measurement given the current estimator state, and S is the innovation covariance, or the covariance of the expected measurement distribution (Equation 5.18).

$$D = \sqrt{(z - \hat{z})^T S^{-1} (z - \hat{z})} \tag{5.17}$$

$$S = H P_p(k) H^T + R \tag{5.18}$$

The Mahalanobis distance is unitless, but it represents the number of standard deviations from the mean. A threshold value can be used to filter out measurements that are too far from the expected distribution. It is important to note that the Mahalanobis distance is directly affected when the process and noise covariances of the estimator are tuned, so effective threshold values can vary. Figure 5.12 shows the results of rejecting outlying lighthouse measurements based on their Mahalanobis distance. In this experiment, IMU data (with a 30 Hz update rate) was gathered from the SCμM development board in conjunction with lighthouse triangulation measurements. An Optitrack motion capture system was used to obtain ground truth pose data for the experiment. A linear Kalman filter was used to estimate the position and velocity of the SCμM board, while the attitude estimation was replaced with the ground truth attitude data in order to isolate the performance of the six-state position estimator. The IMU measurements were transformed from the body reference frame to the global reference frame using the ground truth attitude data. Many of the outlier measurements in this experiment were at the same value. This could potentially be due to firmware processing errors or communication errors. It has also been observed that these errors occur much more frequently when the motion capture system is operating, which may indicate that the system's infrared camera exposure LEDs could be interfering with the lighthouse infrared pulse timings, despite synchronization efforts.

5.7 Integrating Attitude Estimation

The sensor fusion methods discussed so far have required ground truth attitude information in order to work, which will not be available in most SCμM applications. These models showed that IMU data could be incorporated into lighthouse measurements for velocity and position tracking. However, an attitude estimator must be developed for a fully functioning SCμM tracking system.

Attitude estimation was implemented using the Madgwick filter [85], which in this implementation utilizes the gravity acceleration vector and gyroscope measurements to

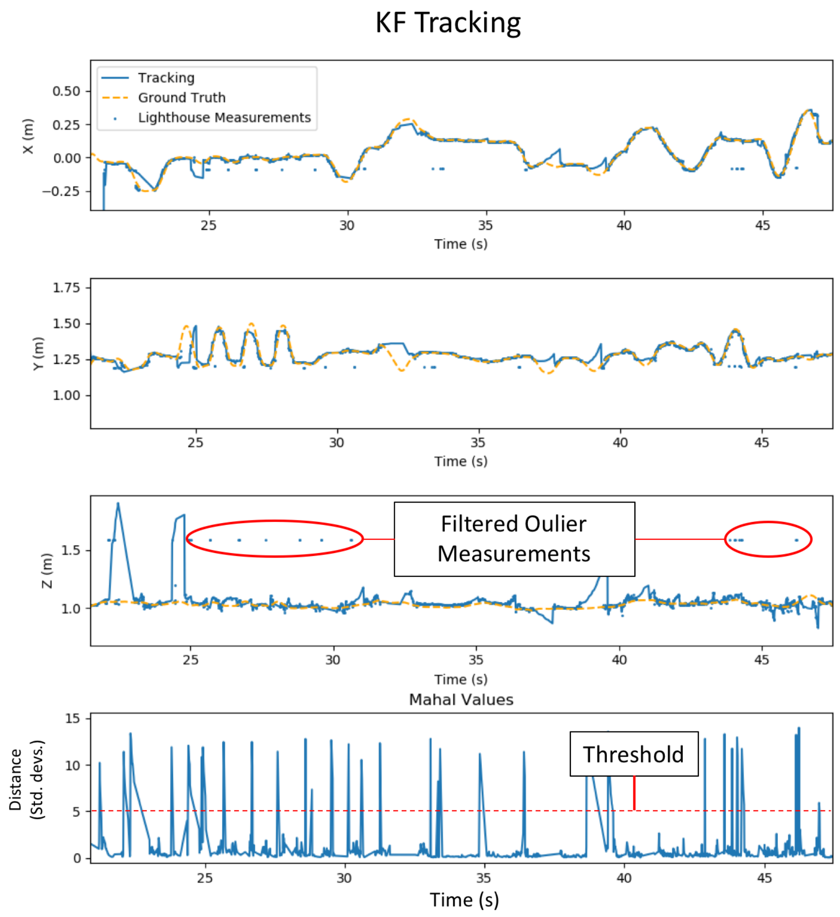


Figure 5.12: Outlier rejection using Mahalanobis thresholding. The threshold used to reject oultiers in this experiement was 5.

estimate attitude. When only accelerometer and gyroscope measurements are used, one attitude axis cannot be estimated without drift. In this case, the yaw euler angle axis, which is about the global z axis, will drift. The filter can also incorporate magnetometer measurements, which will eliminate this drift. However, the drivers required to use the ICM20948's magnetometer have not been developed yet. The SCμM transmitted packets containing IMU and lighthouse data to a processing computer. The nominal IMU data rate was 120 Hz.

A linear Kalman filter was used to estimate position and velocity. A yaw error state was added to this Kalman filter, which is intended to estimate and correct the drift error that occurs in the Madgwick filter's yaw angle estimation. The inclusion of this state required the implementation of an Extended Kalman Filter, which essentially linearizes the nonlinear dynamics of the yaw error state so they are compatible with the Kalman filter equations. The attitude estimate was used to remove the gravity vector from the accelerometer measurements and to transform them into global coordinates from body coordinates. The Mahalanobis

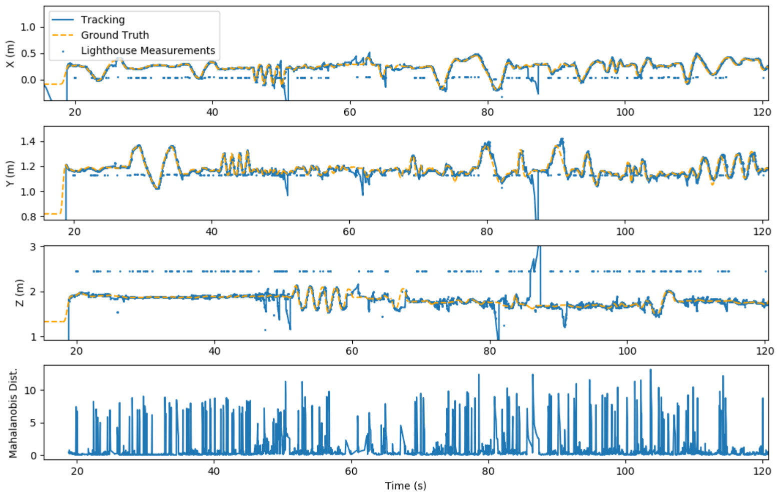


Figure 5.13: Position tracking results from sensor fusion experiment. The bottom plot shows the Mahalanobis distance of each measurement the estimator's predicted measurement distribution. Outlier measurements with a Mahalanobis distance greater than 2.5 were ignored by the tracking algorithm, the results of which can be illustrated by the lighthouse measurements plotted in the position graphs.

distance was used to detect outlier lighthouse triangulation measurements. Figure 5.6 shows the architecture of this filter. The standard deviation of the measurement noise was tuned to 5 cm. The position process noise standard deviations were set to 3.2 cm, the velocity process noise standard deviations were tuned to 7 mm/s, and the yaw drift process noise was tuned to .001 radians.

The filter performance was evaluated in an experiment where a SCμM development board with an IMU was tracked via lighthouse localization and an Optitrack motion capture system. As in previous work [65], the motion capture system was synchronized to the lighthouse system to avoid interfering with the lighthouse localization pulses. Figure 5.13 shows the position tracking performance of this algorithm. Table 5.1 summarizes the RMS tracking accuracy from this experiment. Figure 5.13 also illustrates the effectiveness of outlier rejection using the Mahalanobis distance, which improved the accuracy of the tracking system.

The attitude estimation performance of the Madgwick filter can be seen in Figure 5.14. For both pitch and roll, the attitude estimator was effective, as the estimator could use information about the direction of the gravity vector to eliminate drift. However, the lack of magnetometer data is manifested in the yaw drift error, where the estimator did not have enough data to eliminate drift. Unfortunately, the yaw angle estimator in the Kalman filter did not successfully eliminate these drift errors (Figure 5.14). Errors in the yaw angle estimation decrease the performance of the position tracking. Implementing the magnetometer driver on SCμM and including compass heading data in the Madgwick filter should eliminate

Position	**RMS Error (cm)**
X	7.7
Y	5.1
Z	18.0

Table 5.1: Positioning Accuracy of Lighthouse-IMU Sensor Fusion of the experiment shown in figure 5.13

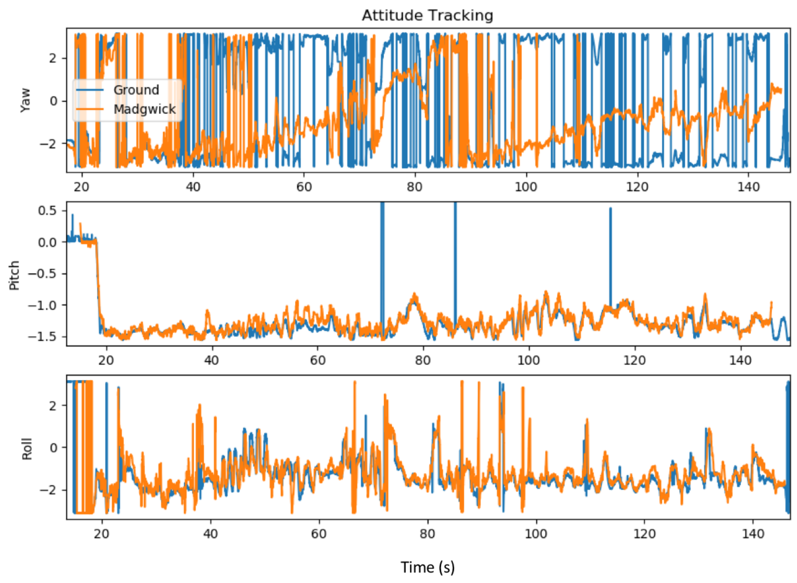


Figure 5.14: Attitude tracking performed by Madgwick filter. Pitch and roll euler angles were successfully estimated without drift by the Madgwick filter. The filter could not provide a drift-free estimate of the yaw euler angle, which is about the z axis and the gravity vector. Providing magnetometer data to the Madgwick filter would fix this issue.

this yaw angle drift.

Chapter 6

Cooperative Lighthouse Localization for Resource-Constrained Robots

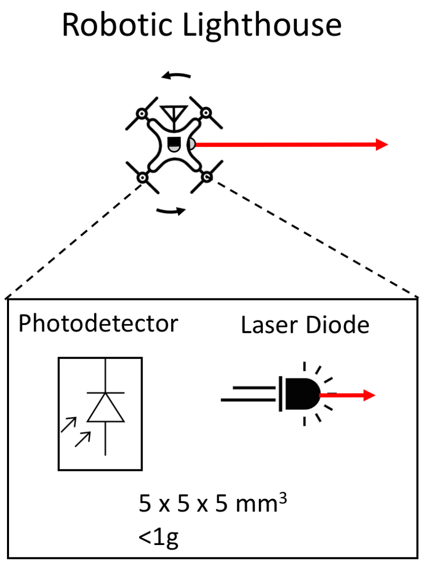
6.1 Introduction

As detailed earlier in this dissertation, lighthouse localization is a promising localization method for resource-constrained microrobots that leverage SCμM. In a real-world application, there will likely be many microrobots working together as a team to perform a task, communicating via wireless mesh network. There may not be stationary lighthouse base station infrastructure available in these situations, so vanilla lighthouse localization may be difficult. The following chapter proposes a cooperative lighthouse localization system, leveraging wireless mesh networking, that could be used by teams of microrobots to self-localize. In this system, robots act as mobile lighthouse beacons, equipped with laser plane generators and rotating to perform lighthouse azimuth sweeps. Figure 6.1 provides an illustration of a robotic lighthouse used in this cooperative localization system.

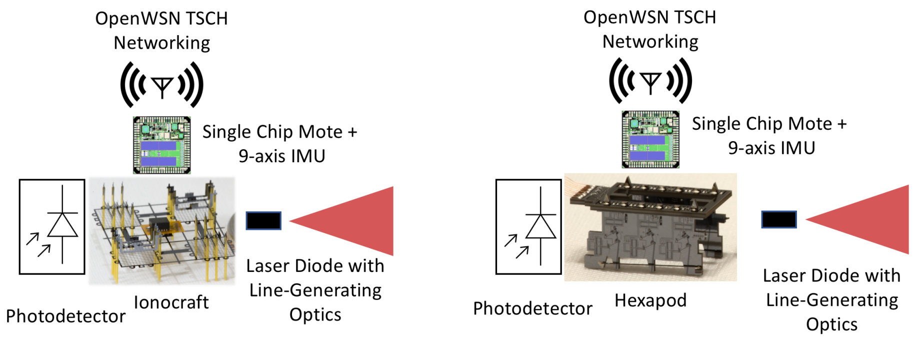
6.2 Related Work

Robotic wireless sensor networks, defined as networked multi-agent robotic systems intended for sensing the environment [86], have many potential applications including search and rescue [87], wireless sensor network deployment [88, 89], and underground mine exploration [90]. These systems combine the advantages of low-power mesh networking with the mobility of robotics. For example, a persistent, low-power wireless sensor network (WSN) could be used as localization reference points for a robot traversing the network area. However, these low-power WSN systems have limited computational resources, which create unique challenges for their use in localization.

Localization is important for the coordination and control of robotic systems. Furthermore, localization is useful for many applications of wireless sensor networks that require geospatial information in addition to sensor data [91, 92]. While modern global positioning systems



(a)


OpenWSN TSCH Networking
Single Chip Mote + 9-axis IMU
Photodetector
Ionocraft
Laser Diode with Line-Generating Optics
OpenWSN TSCH Networking
Single Chip Mote + 9-axis IMU
Photodetector
Hexapod
Laser Diode with Line-Generating Optics

(b)

Figure 6.1: Microrobotic lighthouse setup. Microrobots, such as the hexapod [7] or ionocraft [5] shown in this figure, could act as lighthouse beacons by rotating while equipped with laser diodes and line-generating optics. In this system, the microrobots would have to keep track of their heading as they rotate, and periodically transmit it to other robots that are using the beacon laser to localize.

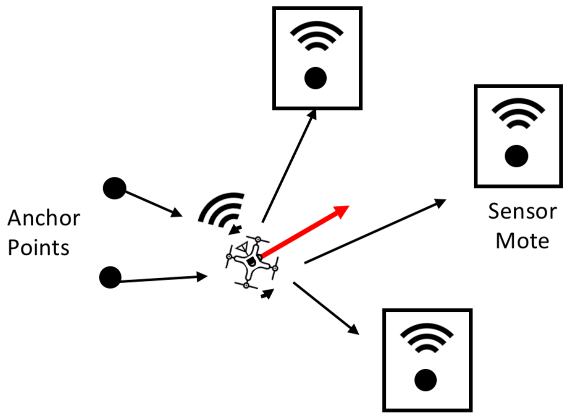


Figure 6.2: Schematic of proposed distributed lighthouse self-localization. The quadrotor can acts as a lighthouse base station by rotating in order to provide and obtain relative headings distances between itself and wireless sensor nodes. In this system, the quadrotor can localize itself based on bearing measurements relative to anchor sensor nodes with node locations, and it can provide bearing measurements to sensor nodes with unknown locations, which allows them to localize themselves

(GPS) function well in most outdoor locations, they do not function reliably in most indoor environments. Some multi-agent robotic control methods also use infrastructure like VICON for localization, where high resolution cameras are placed around an environment to triangulate an accurate pose estimation of various objects, which requires significant infrastructure overhead [93] and is generally not easily scalable. Many robotic localization systems rely on simultaneous localization and mapping (SLAM), which requires enough computational power to process keypoints from images taken of the robot's surrounding environment [94], as well as register landmarks globally to perform loop closure. These systems are suitable for robots with large enough computation power, and even then, rely on feature-rich environments. Smaller, low-power, low-cost robots usually don't have the computational resources to process full images, and they can be spread over a large area, potentially out of range of stationary localization infrastructure.

Many localization methods for low power WSN nodes have been developed, including ultrasonic, RF, and optical methods. These localization methods use various modalities to obtain measurements of distance, angle of arrival, time of flight, or time delay of arrival. Received signal strength intensity (RSSI), a value that is easy to measure using RF communication, only provides accuracy on the order of a few meters[91, 92]. One localization method, lighthouse localization, could be particularly useful for wireless robotic sensor networks due to low computational overhead and spatial resolution on the order of a centimeter [91]. Lighthouse localization [63] was originally intended for smart dust [95] and uses laser sweeps to accurately localize computationally resource-constrained sensor motes. The advantage

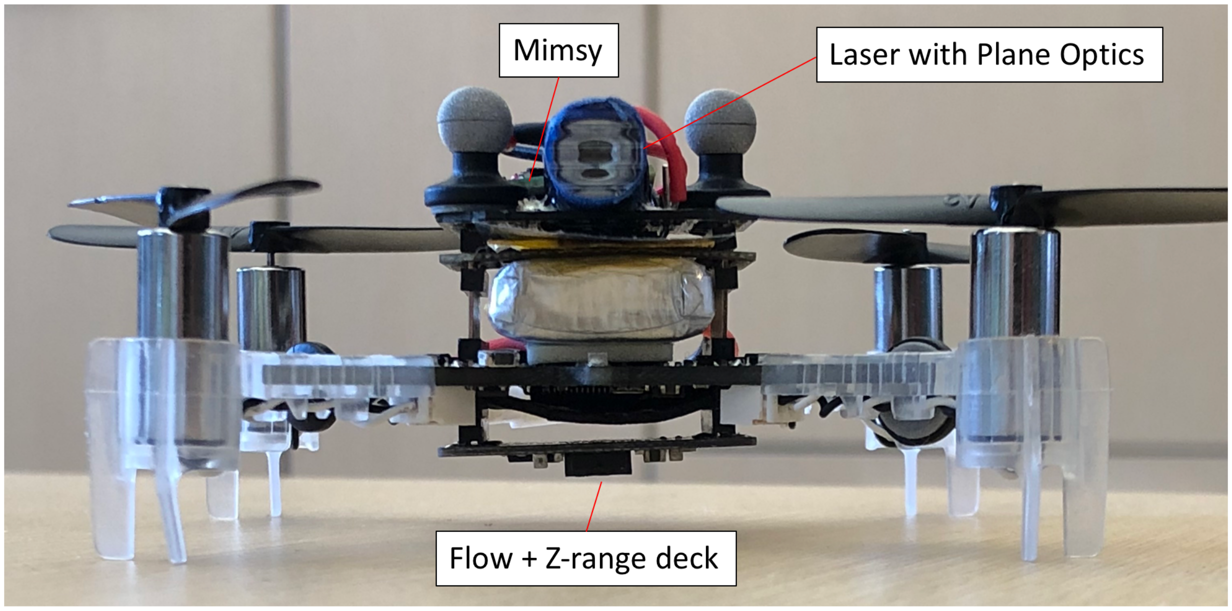


Figure 6.3: Crazyflie minidrone platform with a MIMSY, 850nm laser, plane-generating optics, and an optical flow deck

of this system is that the self-localization requires little computation on the sensor motes, because the only calculation required is a time-to-orientation mapping. This localization method also scales well with the number of sensor motes in the system. HTC Vive's 2015 release of their Lighthouse Base Station, used as the basis for their virtual reality system, led to a resurgence in the use of lighthouse localization in recent years. Lighthouse localization schema like those of HTC allow for easily scalable, low-cost, low-latency, and high-precision pose estimation, providing objects with constellations of infrared-sensitive diodes on them with the ability to recover their 6 degree of freedom pose at update rates of up to 60 Hz [96]. Furthermore, lighthouse localization has been demonstrated to accurately localize the 2mm x 3mm Single Chip μMote (SCμM) wireless SoC [25, 65]. The 5 mg SCμM, as described in chapter 4, requires no external components outside of a battery and antenna, and represents the state of the art in miniaturized wireless sensor nodes. However, lighthouse localization still requires stationary lighthouse beacons to localize sensor motes, which would constrain the spatial range of the robot.

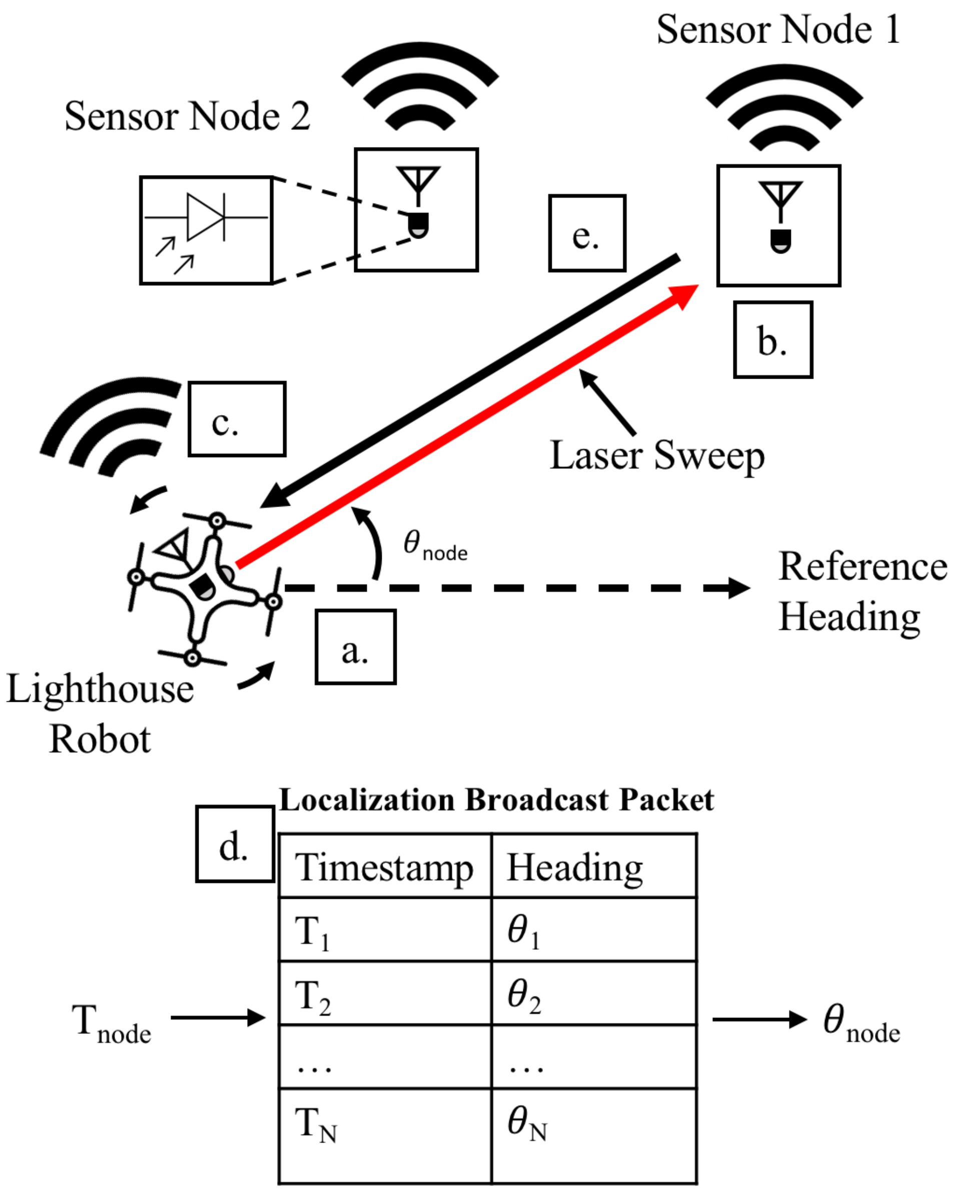
Relative measurements between robots have been utilized in distributed and cooperative localization methods for teams of robots [97, 98, 99, 100, 101]. These methods determined relative positions and orientations between robots using a variety of methods such as sonar distance estimation for teams of robots [98], cooperative inchworm localization with optical pose estimation [97], and range-only SLAM for multiple micro-air-vehicles [100]. Relative measurements could also be used between wireless sensor nodes and robots for localization. Indeed, numerous relative bearing-only measurements systems have been formulated that allow mobile robots to localize stationary and non-stationary targets [102, 103, 104]. Bergbreiter et al. [105] demonstrate a particularly applicable system for localizing teams of resource-constrained robots with relative bearing measurements. They use robot-mounted beacon LEDs and fisheye lenses to obtain optical bearing measurements between robots. The system, including a custom integrated circuit designed to receive the light signals and a microprocessor for processing, would have a footprint of around 3 cm x 5 cm and has a light computational load required for localization.

Cooperative localization systems like these have also been demonstrated successfully in some aerial vehicle teams. For example, Roberts et al. [106] developed a 15 cm diameter 3D relative positioning sensor for localizing team of aerial robots. This sensor was used to enable a swarm of flying robots to successfully navigate through an indoor environment using relative position measurements [107].

We propose a bearing-only measurement system for drones that utilizes laser lighthouse localization and heading measurements obtained from an inertial measurement unit (IMU). Similar to the system in Bergbreiter et al. [105], robots will carry both laser beacon lights and photo detectors to obtain bearing measurements relative to other robots. Compared to the 120 gram, 15 cm diameter positioning sensor in [106], the hardware required for the proposed system is both smaller and lighter. This system will has a low computational load, since anchor points only need to calculate a simple mapping between time and relative heading. When a quadrotor acts as a lighthouse beacon, it turns on a laser and rotates while simultaneously keeping track of its heading angle vs. time in a lookup table. All wireless sensor anchors keep track of when they receive laser pulses. The lighthouse drone then wirelessly broadcasts its saved heading angle vs. time look up table, and all other sensor nodes can interpolate their relative heading to the lighthouse, and thus determine location information. The lighthouse drone can then receive these bearing-only measurements from the anchor nodes, and incorporate them into its state estimator to improve its position estimate. This chapter demonstrates the use of quadrotor-lighthouse bearing measurements, relative to wireless sensor nodes of known location, to estimate the position of a quad rotor. This paper also presents simulations that show that this system should be able to provide relative bearing measurements that can localize wireless sensor nodes with unknown locations.

6.3 Robotic Lighthouse Localization System

The quadrotor lighthouse localization system consists of a quadrotor equipped with a low-power IR laser diode and wireless sensor node, and wireless sensor nodes equipped with a circuit containing a high radiant sensitivity BPV22NF photodiode with a daylight blocking filter, transimpedance amplifier, passive filter, and gain stage. The wireless sensor node hardware system is comprised of the Micro Inertial Measurement System (MIMSY) [40], a wireless sensor mote containing a Cortex-M3 microprocessor, 9-axis IMU, and 802.15.4 transceiver. The quadrotor's 1 mW laser diode is equipped with line-generating optics, which produces a planar laser beam. These components are used to produce the lighthouse sweep,



Timestamp	Heading
T_1	θ_1
T_2	θ_2
…	…
T_N	θ_N

Figure 6.4: a). The lighthouse robot rotates while recording orientation-timestamp information from its attitude estimator. b). Sensor 1 detects laser sweep using its photodetector and records the network-synchronized timestamp of the laser detection event at T_{node}. c). Lighthouse robot periodically broadcasts timestamp-orientation mapping, which other robots use to localize themselves. d). Sensor 1 uses the received time-orientation broadcast and T_{node} to calculate its relative heading θ_{node}. e). Sensor 1 sends the relative heading back to the lighthouse robot.

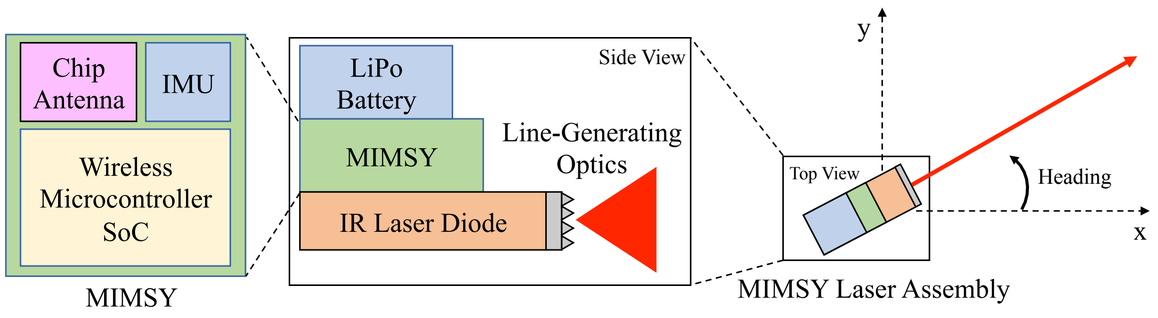


Figure 6.5: System Block Diagram

record the lighthouse heading measurements, and transmit the timestamp-heading tuples (Figure 6.4)[1].

Figure 6.4 illustrates the operation of the quadrotor lighthouse localization system. As the lighthouse robot rotates, it records its heading with respect to magnetic north and then broadcasts its previous timestamped headings to every mote in network range. The other sensor nodes in the system use the time at which they detect a laser pulse and the received heading packet to determine their heading relative to the lighthouse robot.

The wireless sensor hardware was integrated into a Bitcraze Crazyflie 2.0 quadrotor using a UART interface. The Crazyflie 2.0 performed all of the real-time state estimation calculations and periodically updated the MIMSY with X, Y, and yaw estimates via UART at 40 Hz. The MIMSY communicated any lighthouse measurements that it received to the quadrotor using this UART interface. Figure 6.3 shows the full MIMSY, laser, and Crazyflie system. The Crazyflie system also included an optical flow deck, which contains an optical flow sensor and z-ranging infrared sensor. These provided velocity and Z measurements to the quadrotor. An Extended Kalman Filter (EKF) included in the Crazyflie firmware was used to provide heading information to the MIMSY for the lighthouse measurements.

Synchronized timekeeping is imperative for this proposed quadrotor lighthouse method to function; every robot in the network must have a synchronized sense of time in order to calculate their relative headings to the lighthouse robot. Fortunately, many wireless sensor network architectures already maintain a synchronized network-wide timebase during normal operation [108, 109]. Specifically, IEEE 802.15.4E networks use Time-Synchronized Channel Hopping (TSCH), which improves network reliability and reduces power consumption [108][110]. A synchronized sense of time is maintained between all motes in the network and is denoted by the absolute slot number (ASN), which indicates the current timeslot number of the network. Motes in the network communicate, wake-up, or sleep, depending on the current timeslot and where it resides within the network schedule. The ASN only provides time resolution according the timeslot length, which is 10 ms in our network implementation; we improved this resolution by using the motes onboard crystal timers to get 30.5μs resolution when timing lighthouse-related events.

[1] Drone icon adapted from icons made by Freepik from www.flaticon.com.

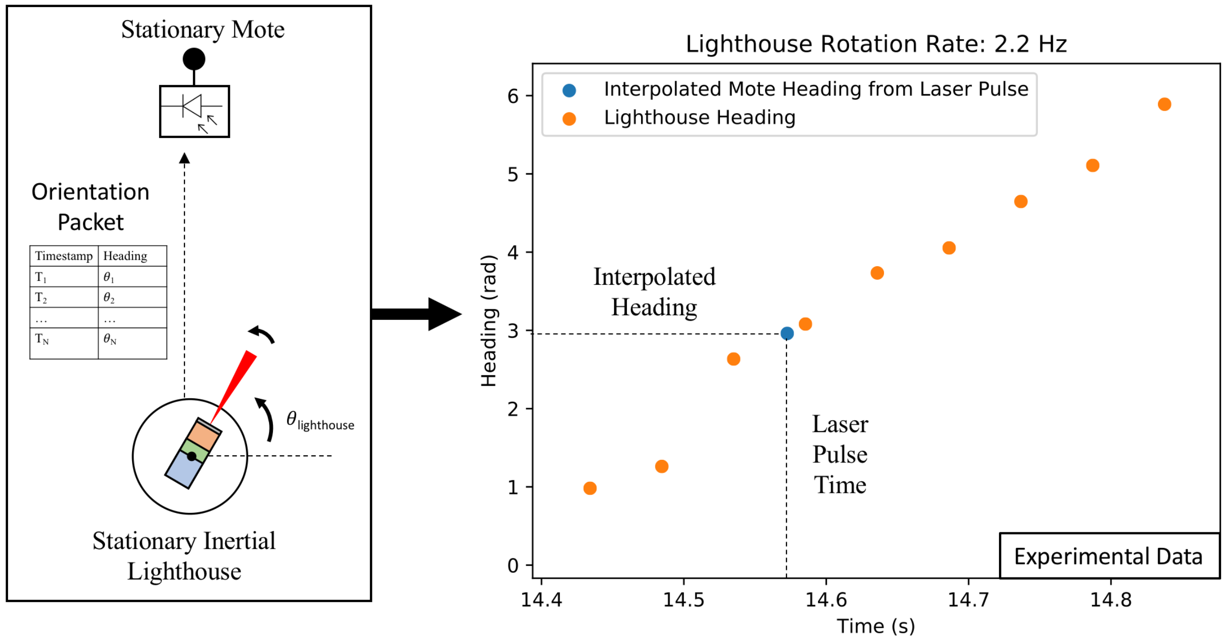


Figure 6.6: Calculation of heading from laser pulse timing. Once a node detects a laser pulse and receives a localization EB, it can calculate its heading relative to the lighthouse node by interpolating the surrounding lighthouse time-heading tuples and looking up the laser pulse time. This data was obtained during an experiment with a fixed-position lighthouse mote rotating and a stationary self-localizing mote. This heading data was produced with the onboard magnetometer on a MIMSY that was located indoors.

The wireless sensor network implementation we used was OpenWSN, which is a full-stack, standards-compliant implementation of a wireless sensor network based on the IEEE 802.15.4E standard [42]. OpenWSN can run on low-power, resource-constrained devices and provides routing, scheduling, transport layer, and internet integration. In addition to a synchronized timebase, OpenWSN provides peer-to-peer communication in the form of Enhanced Beacons. Enhanced Beacons (EBs) are periodically broadcast by all motes in an OpenWSN network and are used to advertise the network so new nodes can join [111]. Information Elements (IEs) are information containers within EBs that can store various types of network-related information. For use in the distributed lighthouse system, we created location and time-heading tuple IEs, which enabled the transmission of the information necessary for motes to perform localization from a laser pulse. Figure 6.7 shows the formatting of the EB packet that was broadcast by beacon robots. These beacons were redundantly retransmitted three times in order to improve the reliability of the communication, which leveraged reliability improvements possible with TSCH.

When a mote detects a laser sweep as a pulse from its photodiode, it records the pulse's timestamp. It then waits until it receives a localization EB. Once a localization EB and laser

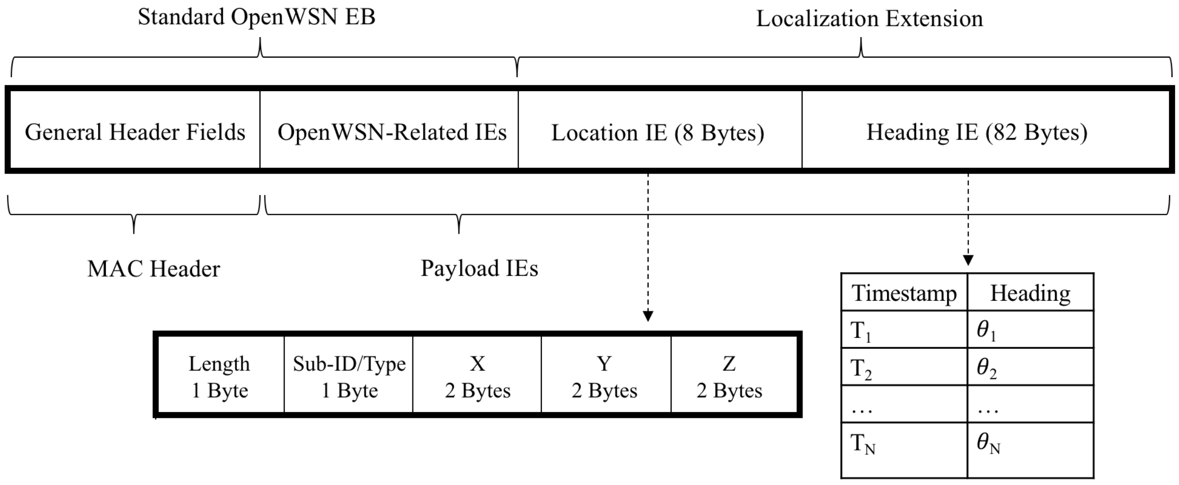


Figure 6.7: OpenWSN motes periodically broadcast packets called enhanced beacons (EBs), which contain network-related data. EBs contain Information Elements (IEs), which are containers used to store specific types of payload. The lighthouse mote uses EBs to broadcast time-heading tuples.

pulse have been received, the mote calculates its heading. During the calculation procedure, the mote searches the localization EB tuples until it finds a time-heading tuple before and a tuple after the recorded laser pulse time. The mote then linearly interpolates between these two points and calculates the heading that corresponds to the laser pulse time (Figure 6.6).

In this work, the wireless sensor nodes used were MIMSYs; however, the underlying robotic lighthouse system is compatible with the Single-Chip Mote (SCM), which is a monolithic wireless system-on-a-chip (SoC) that requires no external components [25, 65]. This mote contains an IEEE 802.15.4 radio, allowing it to support OpenWSN, which provides the networking backbone for this measurement system. Furthermore, the SCM contains an optical receiver that has been shown to provide accurate localization performance in a commercial HTC lighthouse system, as discussed in chapter 4 [65]. This compatibility will allow cubic millimeter sensor motes to act as localization anchors and eventually be localized through the robotic lighthouse measurement system.

6.4 Wireless Reliability

The proposed localization framework is dependent on reliable communications, especially with its relatively slow measurement rate. As the measurement communication process requires a successful round trip communication between the lighthouse robot and the anchor points, which consists of two successful transmissions. For example, if the packet delivery ratio, the ratio of transmitted packets that are successfully received, of the link is 80%, then only 64% of round-trip measurement communications will be successful, effectively reducing the update rate of measurements in the system. Unfortunately, the low power 2.4 GHz radios and suboptimal chip antennas used in this hardware system and many other robotics systems are susceptible to path loss, interference, and multipath. Indeed, PDR measurement experiments for our hardware taken in the experiment volume were not optimistic (Figure 6.9). For this specific hardware, the PDR up to 3 meters was roughly 75%, while the PDR at distances between three and six meters was 45% and 65%.

Fortunately, wireless sensor networks use similar radio hardware that yields lossy communications, and significant effort has gone into developing methods to cope with lossy communications in wireless sensor networks. One successful technology for improving reliability is time-synchronized channel hopping (TSCH) [108], which helps networks overcome multipath effects and in-channel interference. Indeed, some industrial wireless sensor networks using TSCH and packet retransmissions can obtain up to nine nines of reliability (99.9999999% of transmissions completed successfully) [112, 113]. In a network using TSCH, all the nodes follow a synchronized schedule dictate which channel the nodes' radios should be using at which time subdivision. This schedule is called the TSCH matrix and is illustrated in Figure 6.8.

In order to overcome the low PDRs inherent in the low power radios and the experiment setting, packet retransmissions with TSCH were used. Without TSCH, rapidly retransmitting packets is ineffective at improving reliability because retransmissions occur on the same channel frequency, and any disturbance is likely to affect retransmissions as well. Conversely, retransmitting rapidly on different channels can provide greater reliability because each channel is independent [114, 115]. In this work, each broadcast consisted of three redundant transmissions of the same data packet. With the timeslot of 10 ms in this network, each redundant packet should be sent 10 ms after the previous transmission. Figure 6.9 shows this method providing >98% reliability up to distances of 3 meters, and 80%-95% reliability in distance between 3 and 6 meters. This reliability improvement likely wouldn't have been possible if used in a non-TSCH network.

6.5 State Estimation

Quadrotor Localization with Bearing Measurements

State estimation and sensor fusion is required in this system in order to provide timely state information for the quadrotor. The estimator used in this work is based on the Extended Kalman Filter derived in [82, 83] and implemented by Bitcraze in the Crazyflie firmware, which utilizes IMU data for state propagation and can incorporate a variety of measurement updates. The robotic lighthouse measurements were incorporated into the state estimator as measurements. Each measurement contained the location of the sensor node anchor point the measurement came from and the relative bearing measurement to that anchor point.

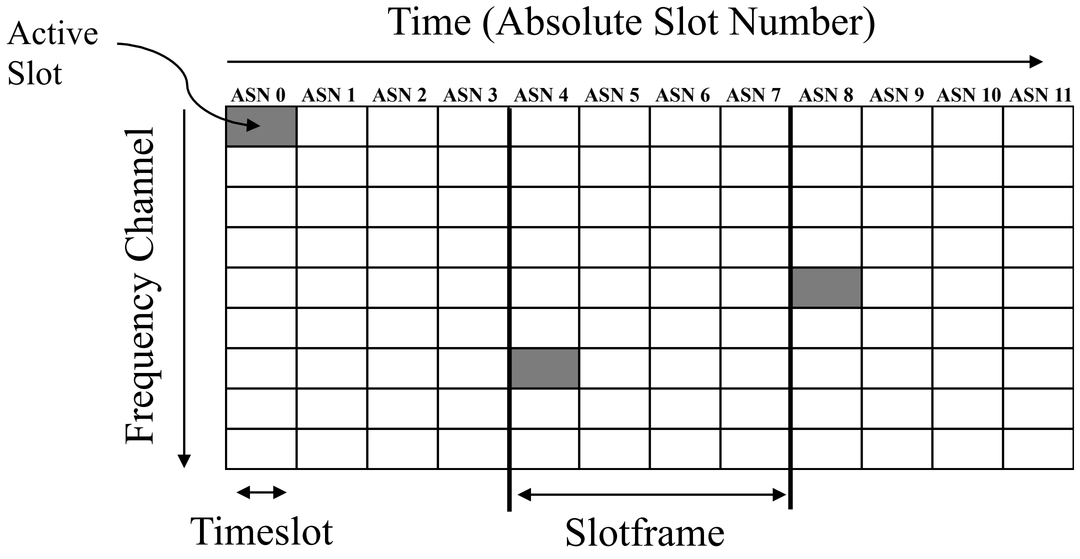


Figure 6.8: Time-synchronized channel hopping (TSCH) matrix for OpenWSN. Timeslots in the TSCH matrix correspond to points in time and frequency and can be scheduled as active and inactive. Active slots can be used for transmitting or receiving packets, while inactive slots are used for saving power. All motes in a TSCH network must maintain a common synchronized sense of time in order for this network architecture to operate. The absolute slot number (ASN) reflects the current network-wide time in units of timeslot, which are 10 ms long in this application. This universal timebase enables infrastructure-free lighthouse localization.

The Crazyflie estimator has nine states (Equation 6.1), where x, y, z denote position, vx, vy, vz denote velocity and $D0, D1$, and $D2$ denote attitude error [82, 83]. Attitude is stored in a separate reference attitude, which is periodically updated using the attitude error states.

$$\vec{x} = \begin{bmatrix} x & y & z & vx & vy & vz & D0 & D1 & D2 \end{bmatrix}^T \tag{6.1}$$

The measurement model for a lighthouse relative bearing measurement between a rotating quadrotor, with 2D location (x, y), and a stationary anchor point, with 2D location (x_a, y_a), at timestep k is shown in equation 6.2, where w represents gaussian noise. The linearization of this model with respect to the state model is shown in equation 6.4, where the only nonzero entries correspond to the x and y states. In equation 6.4, R is the distance between the anchor point and the quadrotor's estimated X,Y locations, and θ is the predicted bearing measurement.

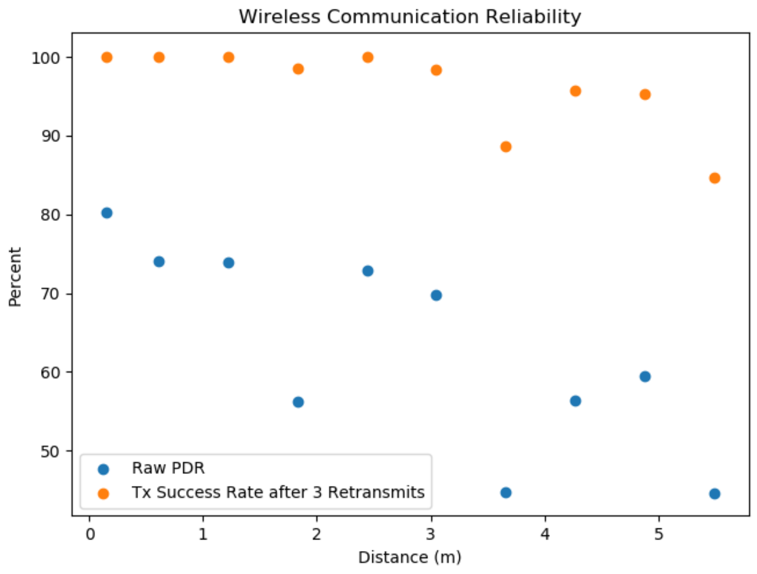


Figure 6.9: Reliability of robot to anchor beacon transmissions within OpenWSN. Each broadcast from robot to anchor consisted of three redundant retransmissions of the broadcast's packet. Packet delivery rate (PDR) was calculated as the ratio of all packets (including redundant retransmissions) that are successfully received by the anchor, which is also approximately the success rate of a broadcast using not using redundant transmissions. Transmission success rate was the percent of broadcasts where at least one of the three retransmitted packets were received. RF interference within the experiment space decreased the packet, which is evident in the low PDRs at short distances. Retransmitting packets three times improved the success rate of the transmissions.

$$h_k(\hat{x}_p(k), w(k)) = atan2(\frac{y_a - y(k)}{x_a - x(k)}) + w \tag{6.2}$$

$$H = \frac{\partial h_k(\hat{x}_p(k), 0)}{\partial \vec{x}} \tag{6.3}$$

$$H = \frac{1}{R}\begin{bmatrix} sin(\theta) & -cos(\theta) & 0 & \dots & 0 \end{bmatrix} \tag{6.4}$$

$$R = \sqrt{(x_a - \hat{x}_p)^2 + (y_a - \hat{y}_p)^2} \tag{6.5}$$

$$\theta = h_k(\hat{x}_p(k), 0) = atan2(\frac{y_a - y(k)}{x_a - x(k)}) \tag{6.6}$$

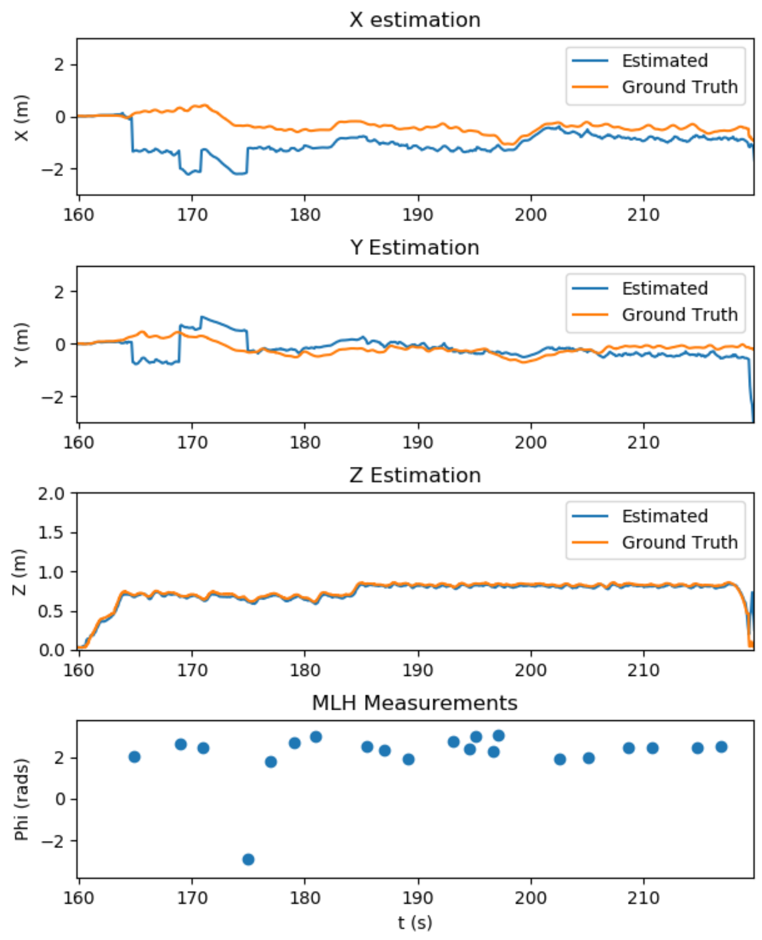


Figure 6.10: X, Y, and Z state estimates during a quadrotor lighthouse experiment. Initial lighthouse bearing measurements were heavily weighted due to high initial state variance (see Figure 6.11). This is illustrated by the large discontinuities in tracking between 165 and 175 seconds, which correspond to lighthouse measurement updates. As the Crazyflie received more measurements from the sensor nodes, the state variance decreased (Figure 6.11), the measurements were less heavily weighted, and the estimator started to recover and track with less error. The bottom graph shows the received lighthouse measurements. The Z state estimation was very accurate and can be attributed to Bitcraze's stock state estimation with optical flow deck, which includes an infrared range sensor. The bottom graph shows the quadrotor MIMSY lighthouse measurements (MLH) received by the quadrotor.

This estimate was incorporated into the Crazyflie's EKF, and the measurement update was performed each time a quadrotor lighthouse measurement was detected. All of the state estimation was computed onboard the Crazyflie's microprocessor.

6.6 Experimental Results

Experiments were performed in a motion capture room using an OptiTrack system for capturing ground truth information with sub-millimeter accuracy. In this experiment, the Crazyflie took off and performed rotations using open loop control. Estimation data was streamed off the quad using the Crazyflie's logging capabilities and radio. Two wireless sensor node anchors with known positions were present during the experiment and were used to localize the quad. Before the quad took off, it's state estimator was reset and it's state variance was set to maximum.

The state estimation of the quadrotor during this experiment is shown in Figure 6.10. With the measurement variance set to 0.0025 radians, the first relative bearing measurement of the experiment had a large effect on the X and Y states, which had high initial state variance of 100 m^2. Subsequent measurements helped the estimator recover and reduce its error, which is shown in Figure 6.12. Figure 6.11 shows the estimated X and Y variance, as calculated by the EKF, throughout the experiment. Performance could have also been affected by a firmware bug where equation 6.4 was given the actual bearing measurement instead of the predicted bearing measurement $\theta = h_k(\hat{x}_p(k), 0)$. The lighthouse bearing measurements clearly reduce the estimator's variance estimate throughout the experiment. The yaw estimates broadcast by the rotating drone were obtained from the Crazyflie's attitude estimates, which were based off of the gyroscope measurements from its IMU (Figure 6.13). Over the timeframe of this experiment, gyroscope drift accumulated to a 4 degree error near the end of the quadrotor's flight. For longer experiments or missions, magnetometer data could be incorporated to reduce gyroscope drift. The measurement error is shown in Figure 6.14, which compares the measured relative bearing to the actual bearing between the anchor and quadrotor, as calculated with motion capture ground truth data.

6.7 Wireless Sensor Node Localization Simulation

Currently, this robotic lighthouse localization system can provide bearing measurements for a rotating quadrotor acting as a lighthouse. This same quadrotor should be able to provide bearing only measurements to wireless sensor nodes that have unknown locations, as has been shown in literature [102, 103, 104]. This capability would be useful for situations where a wireless sensor network is deployed and locations can't be easily surveyed, like if many miniature wireless sensor nodes were dispersed by a UAV throughout an area. For example, in Mehta et al. [116], a micro-air-vehicle (MAV) is used to deploy a "breadcrumb trail" of wireless sensor nodes that join a wireless mesh network, where the MAV is effectively deploying its

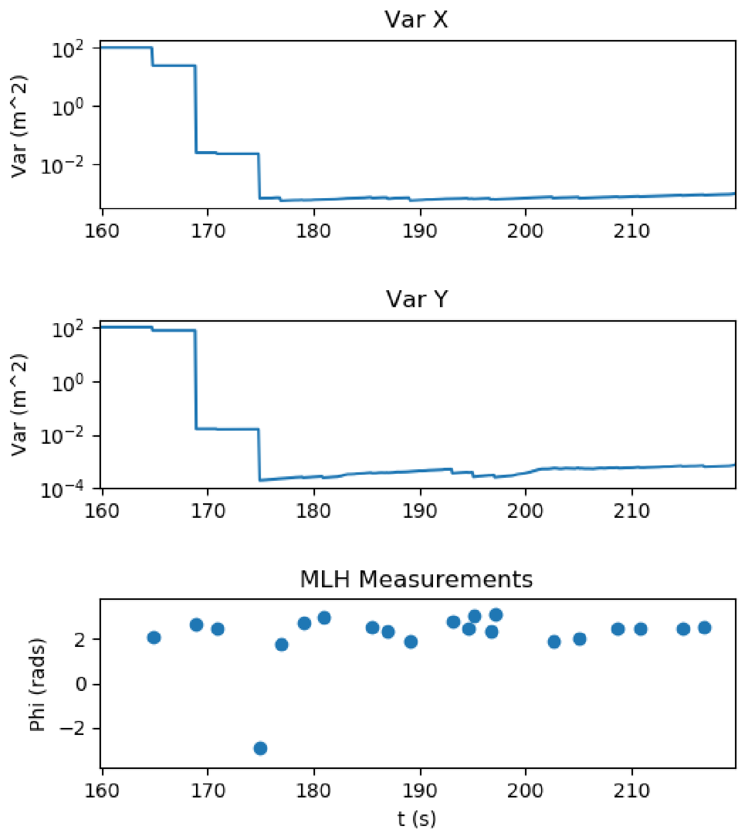


Figure 6.11: Estimator Variance throughout the experiment. The MIMSY lighthouse measurements (MLH) decreased the estimated variance of the x and y states, as expected

own networking infrastructure. These nodes were then used as wireless infrastructure to relay commands to the MAV and transmit video signal from the MAV to a base station. With the presented robotic lighthouse localization system, these node could also act as localization infrastructure for the MAV. The combination of wireless mesh networks and robotic lighthouse localization could enable a MAV to self-deploy its own localization and communication infrastructure by dropping wireless nodes as it travels through a novel area.

We formulated the sensor node state estimation model is an EKF that uses the nonzero portion of the measurement model in equation 6.4, modified with reversed signs as the Jacobian is with respect to the two anchor state $\vec{x_a} = [x_a \quad y_a]$. Since the mote is stationary, there is no state propagation step in the estimator. The EKF allows the sensor mote to estimate its state variance, which it could use to determine when its location estimate is good enough to be used as a localization reference node. Figure 6.15 shows the results of a simulation of a rotating lighthouse robot providing bearing-only measurements to an unknown wireless sensor anchor node. Future work will focus on verifying this simulation in

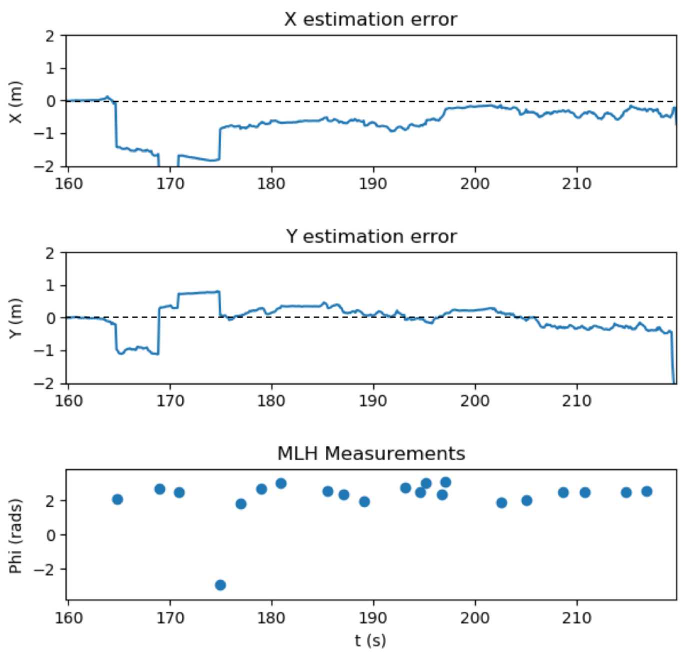


Figure 6.12: Estimation error during the quadrotor lighthouse localization experiment. The quadrotor is performing rotations and receiving bearing measurements from sensor nodes.

experiments, and demonstrating this localization ability.

6.8 Conclusion

The goal of this work was to implement a bearing-only lighthouse localization that could be used by a robot to localize itself from wireless sensor nodes, and eventually be used to localize other wireless sensor node. We achieved this by using attitude estimates from a quadrotor's EKF, the time-synchronization and peer-to-peer communication provided by OpenWSN, low-power IR lasers, photodiodes, and drone maneuvers. We demonstrated the use of these relative bearing measurements in a Crazyflie's onboard EKF to estimate position, and performed simulations that show that this system should be able to localize wireless sensor nodes with unknown location. The laser scan produced by the quadrotor is also detectable by a 10 mg, 2 mm x 3 mm single chip mote, which means this localization method could be used in the future for localizing a network of highly miniature wireless sensor node. Future work will focus on experimentally demonstrating the localization of wireless sensor nodes with unknown position using this measurement system, extending this method to localizing other quadrotors, and incorporating multiple lighthouse robots into the system.

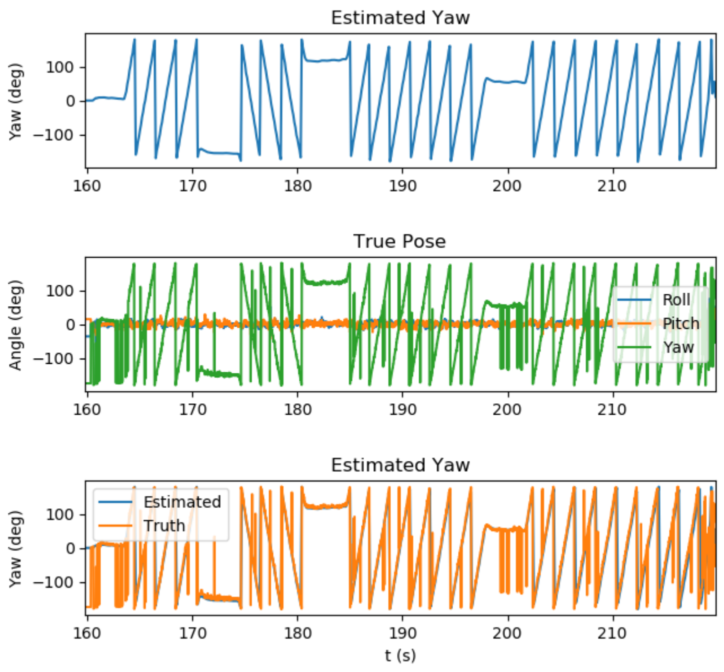


Figure 6.13: Pose estimation during experiment. Pose was estimated using the stock Crazyflie EKF algorithm, which relies on gyroscope measurement data.

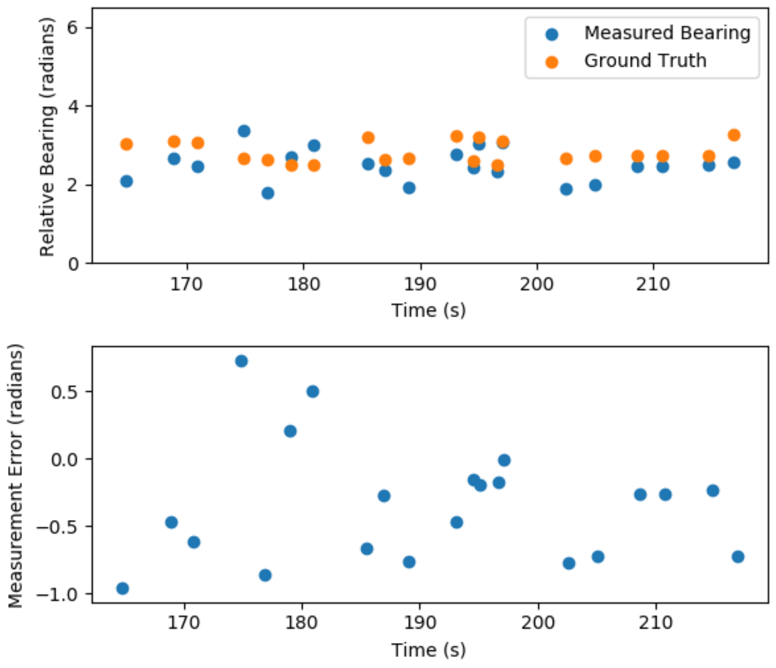


Figure 6.14: Measurement error during quadrotor lighthouse experiment. Top: bearing measurements compared to actual relative bearings determined by ground truth data. Bottom: measurement error. The error bias (mean error) was -0.341 radians, and the standard deviation was 0.431 radians

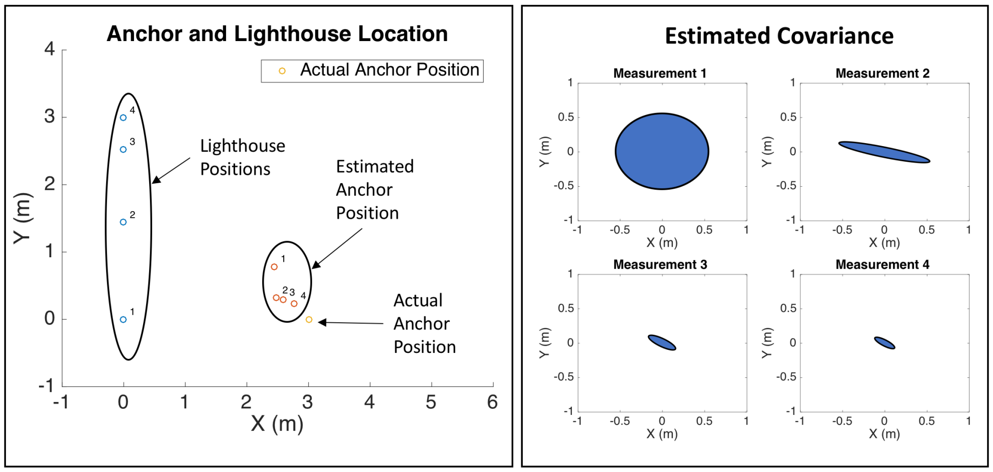


Figure 6.15: Simulation of wireless sensor node localization using lighthouse bearing measurements from a rotating quadrotor that is also moving translationally. In this simulation, the sensor node doesn't know its location and is estimating its location from bearing measurements from the quadrotor lighthouse. As the anchor sensor mote receives more measurements, its location estimate improves and estimated variance decreases.

Part III

Appendix

Appendix A

Practical Assembly and Design Tips

In practice, microrobotic device designs rarely work as expected and many unexpected failure modes manifest themselves in the lab. This chapter attempts to document some of this working laboratory knowledge of MEMS control surfaces, including failure modes of control surfaces and the successful assembly process of the devices.

A.1 Failure Modes

1. The process of gluing out-of-plane protection brackets onto the devices can be very risky. Frequently, silver epoxy drips onto mechanisms or electrical pathways, which freezes the mechanisms of the device or shorts out the device, respectively. This is a recurring problem.

2. Silver epoxy that is used to fasten objects to the device-side of the chips frequently reflows and spreads away from the initial point of application. Eventually, this reflows reaches underneath structures and immobilizes mechanisms. This reflowing occurs when the device is heated to cure the silver epoxy. Figure A.1 shows an example of this reflow behavior on a silicon grid used to prevent out-of-plane motion. Anecdotally, this seems less likely to occur if the silver epoxy is multiple days old and is much more likely when it is fresh. Further anecdotes say that reducing the heat of the curing process can allow the reflow substance to "evaporate" before it reaches critical MEMS mechanisms, but I have not directly observed this process. It is important to note that while silver epoxy is not safe to use on the device side of a MEMS chip, it can be used to epoxy the substrate side of a chip to a surface without the reflowing problem affecting delicate mechanisms on the reverse side of the chip.

3. Proper mounting and alignment of out-of-plane guards is essential. If they are not flush with the pin joints to be protected, out-of plane movement will occur at the joints, causing them to pop out of plane. This was observed multiple times with the first

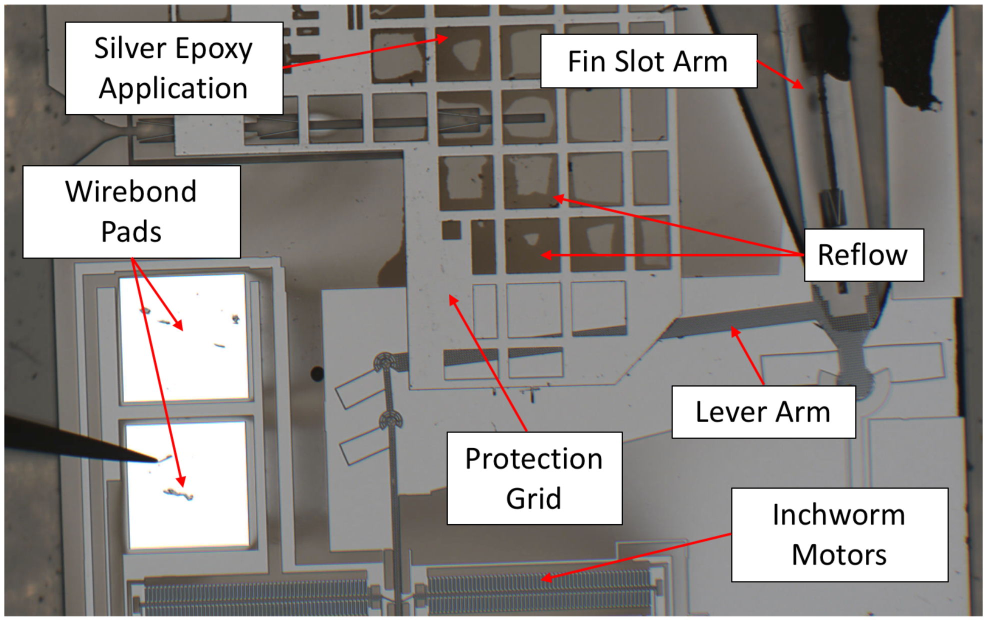


Figure A.1: Example of silver epoxy reflow. Silver epoxy residue can be see surrounding the outline of large portions of the silicon grid, even though the epoxy was only applied on one point of the grid. This wicking is exacerbated when the device is heated to cure the epoxy.

generation of the out-of-plane guard [38], where the central fin slot joint would dislocate out-of-plane when place in the wind tunnel.

4. The process of mounting a control surface onto a glass cover slide, mostly used in [38] using cyanoacrylate glue was fragile and fraught with issues. First, the thin glass cover slides were fragile and could break if too much pressure was applied to them. Second, the cyanoacrylate glue did not adhere well to the smooth surface of the silicon substrate, so the device frequently became dislodged. Finally, the increased temperature of the silver epoxy curing process, if done after a device is glued to the cover slide, can potentially comproise the cyanoacrylate glue.

5. In previous designs, [37] and [38], the solder pads for the electrical wires were too close to the MEMS device and the wirebond pads. Mistaken movements or small tremors would frequently cause soldering irons or wire to accidentally strike the wirebonds and the MEMS device, usually resulting in damage.

6. Wirebonding, in general, is unreliable on a thin flex PCB. First, the flexible PCB substrate tends to absorb ultrasonic energy from the wirebonder, which reduces reliability of the bonds. The flex PCB also doesn't lie perfectly flat on the wirebonding chuck, so solid contact isn't made between wirebonding chuck and PCB, reducing the reliability of the bonding process. Finally, the flexible PCB substrate is pliable enough to allow

movement during wirebonding if the section being wirebonded is too far from the chucks mounting vacuum holes. Adding carbon fiber underneath the bonding sites or specifying flex stiffener underneath the sites is a solution to these problems.

7. Devices were frequently stored in petri dishes to protect them when not in use. Unfortunately, heavy wires (24 gauge) that are inside of the storage petri dish, which are usually electrical wires used to power the devices, can scrape the top of the devices and break mechanisms during storage. The solution is to secure these heavier wires and keep them far from unprotected MEMS structures.

A.2 Control Surface Assembly Process

1. Check that the devices are functional using a probe station.
2. Epoxy (non-conductive) a sheet of thin 0.2 mm thickness carbon fiber (ACP Composites SKU: CFL-TW-0.008-ADH-1) onto the back of the flex pcb in order to increase stiffness and flatness. The sheets of carbon fiber should be placed underneath the wirebond pads plus grounding pad and the connector pads
3. Braid three 40 gauge speaker wires together and tin the tips with solder. This wire thickness is thin enough to be manageable and sturdy enough to not break. This could be done with an intermediate gauge woven wire instead of manually-braided wire.
4. Solder short braided wires to the relevant connector pads
5. Encase wires in non-conductive epoxy (UV or 5 min cure). This should protect the wires from breaking during assembly
6. solder the free end of the wires to a 5-pin header
7. (Optional) Encase header connections in non-conductive epoxy.
8. Put a thin layer of silver epoxy onto the pad. It is imperative to keep this layer thin to avoid silver epoxy from gushing and contacting the front side of the device.
9. Place the MEMS chip onto the ground pad and cure on hot plate for manufacturer-recommended time.
10. Place a silicon plate or out-of-plane prevention buckle over the primary fin joint and then use a thin wire to place silver epoxy on the plate. This piece of silicon will prevent the mechanisms from popping out of plane. Be EXTREMELY careful during this process. Cure for 15 minutes at 100 C

11. Wirebond using 320 mW power on first bond (gold flex pads) and 270 mW on the second bond (silicon pads). A Westbond Model 7476E Wedge-Wedge Wire Bonder was used for this process in [37, 38, 39]. Wirebond from gold pads to silicon pads; the wirebond connection to the silicon pads is much weaker and will not support the wirebonder's spool tension, which is applied during the threading process between the first and second bonds. Metallizing the bond pads on the devices would improve the structural integrity of the wirebonds.

12. Insert fin into fin slot by sliding it in from the side. Once it is partially inserted, tap the end of the fin to gently insert it. Do not continue holding the fin as you push it in.

13. Silver epoxy fin into place. Be careful as any drops of silver epoxy that land on the wrong spot of the device will ruin the device. Use a thin wire (44 gauge or human hair) dipped in silver epoxy to add it. Cure 15 minutes at 100 C

A.3 Silver Epoxy Caveats

While silver epoxy can be a very useful tool for making electrical and structural contact to microrobotic and MEMS devices in some case, it unfortunately has downsides that must be taken into account. As mentioned earlier in this chapter, silver epoxy can cause significant problems during MEMS control surface assembly. Reflowing issues and epoxy application mistakes can greatly decrease the yield and increase the difficulty of the control surface assembly process. While it is certainly possible to use silver epoxy to successfully assemble microrobots, its use is discouraged. The process of assembling robots with silver epoxy is too difficult and too error-prone. This decreases the yield of manufacturing robots, thus greatly increasing the amount of time required to successfully assemble a working robot.

Designing for ease of manufacture is a good method for replacing silver epoxy as an electrical and structural connection method. For example, the MEMS control surface V2 [38] and V3 [39] were designed in tandem with a flex PCB that would allow wirebonding to make electrical connections. With silver epoxy, the assembly process involved manually holding multiple wires in contact with the MEMS device pads and then carefully applying silver epoxy to each pad. After silver epoxy was applied, the wires would need to be held perfectly in place throughout the entire 15 minute curing process; any movement could break the contact and cause a failure. Moreover, a displaced wire could spread silver epoxy on neighboring structures, causing irreparable electrical or structural damage. The codesign of a flex PCB for wirebonding greatly increased the ease and repeatability of making electrical connections to the MEMS control surfaces, compared to the original silver epoxy method.

The assembly process of the MEMS control surface still uses silver epoxy in select steps: fastening the out-of-plane buckle, securing the fin in the fin slot, and attaching the control surface substrate to the PCB ground pads. Silver epoxy risk is somewhat mitigated in the first two cases, where silver epoxy is applied far ($>$1 mm) from vulnerable mechanisms or structures. For example, the out-of-plane buckle was redesigned to move the silver epoxy

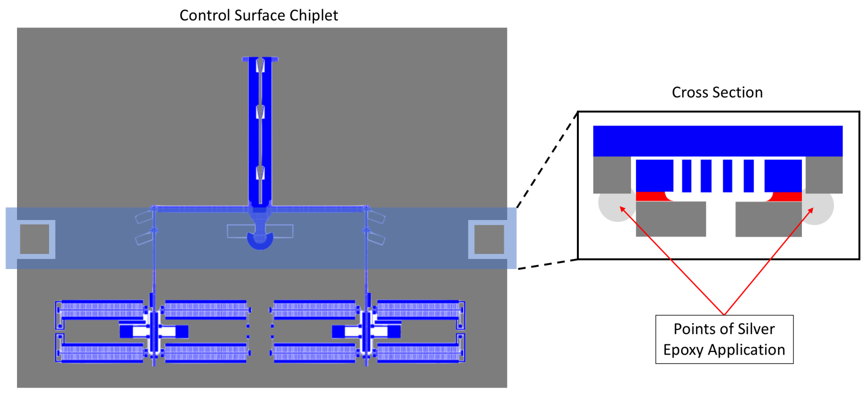


Figure A.2: Out-of-plane protection bracket. Silver epoxy is applied to the edges of the device sustrate, rather than on top of the device layer. This reduces the risk of silver epoxy reflowing and affecting the MEMS devices.

application point from the device face of the chip to the side of the chip, which helped reduce the chance of undesired reflow or wicking from damaging device-side MEMS structures (Figure A.2). The fin slot is physically disconnected by a large trench from any mobile structures, so it is lower risk to place an epoxy dab on the fin to secure it. In both of these cases, the structures are already passively stable in their correct positions and don't need to be held during the curing process. However, this use of silver epoxy still increases the risk of assembly failure and future designs should focus on removing these aspects from the assembly process.

There are several potential methods for removing the need for silver epoxy in microrobotic assembly. For example, wafer to wafer bonding could be used to bond a wafer on top of the MEMS device layer, which would provide protection against out-of-plane motion. A Ziffy MEMS socket [117, 118] could be used to make electrical and mechanical connections with a MEMS microrobot without requiring any wire bonding or silver epoxy.

Appendix B

Scum Lighthouse Implementation Documentation and User Guide

B.1 Hardware Configuration

SCμM requires several configuration modifications in order to detect and process lighthouse pulses, and it will not work with the vanilla configuration found in the SCμM development repo[1]. The optical signals from the lighthouse beacons are detected by the SCμM's integrated optical programmer, which isn't easily accessed by the processor outside of the bootloading process. This is overcome by utilizing GPIO routing. The default general purpose output (GPO) pin bank setting is Bank 0, which routes the data signal of the optical receiver to GPIO 1. In order to use interrupts to detect the lighthouse pulses, GPIO 1 is physically connected to GPIOs 8 and 9. When the GPI bank for these pins is set to Bank 1, GPIO 8 is connected to EXT_INTERRUPT<1>, a level-sensitive active-high interrupt, and GPIO 9 is connected to EXT_INTERRUPT<2>, a level-sensitive active-low interrupt. This configuration allows for the implementation of an edge-sensitive interrupt to time lighthouse sync and laser sweep pulses. Figure B.1 illustrates the GPIO configuration that enables lighthouse localization.

Several other hardware configuration settings are required to get lighthouse localization to work on SCμM. First, the optical receiver is sensitive to the VDDAO voltage level. When VDDAO is too high, it will cause an increased number of glitches in the receiver signal, and if further increased, it will cause the receiver output to rail to VDD. To counteract this, VDDAO needs to be set to the appropriate level, which will vary on a chip-to-chip and, presumably, hardware configuration basis. For example, the QX3 board (individual SCμM development boards are typically identified with Q# or QX# monikers) needs VDDAO to be lowered to at most 0.816 V for the optical receiver to be sufficiently glitch-free. The command **set_ALWAYSON_LDO_voltage(120)** achieved this voltage on QX3; note that in this function, a larger argument will yield a lower voltage. On board QX3, if VDDAO is greater than 0.92V, the receiver will rail high. At 0.87 V, the receiver works, but is very glitchy.

[1]develop branch in https://github.com/PisterLab/scum-test-code

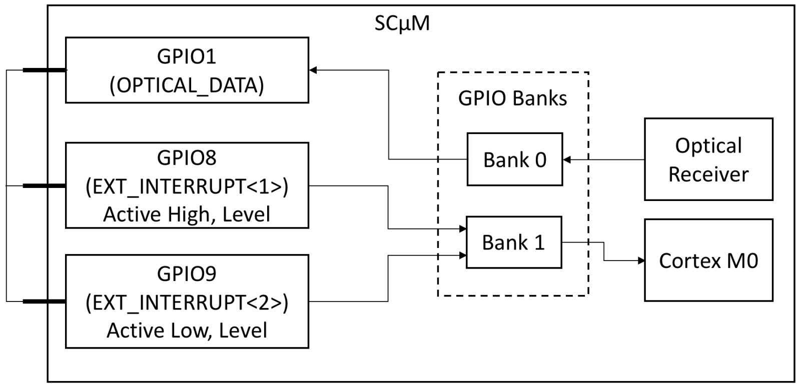


Figure B.1: Block diagram of GPIO configuration required for lighthouse localization implementation on SCμM. GPIOS 1, 8, and 9 must be electrically connected.

The following code snippet illustrates the proper hardware configuration settings to enable glitch-free operation of the optical receiver and use of GPIO interrupts for lighthouse pulse detection.

```
//Reduce VDDAO to improve optical receiver operation
set_ALWAYSON_LDO_voltage(120);

//////////////////////////////////////////////////
//Select banks for GPIO inputs/output configuration

//select input interrupts for gpios 8 and 9
GPI_control(0,0,1,0);

//Select banks for GPIO output: optical rx to GPIOs 0-3
//and cortex-controlled output for the rest of GPIOs
GPO_control(0,6,6,6);

//enable optical pin and external interrupt pins
GPI_enables(0xFF08);

//Enable GPIOs except interrupts as outputs
GPO_enables(0xD0FF);
```

An important note is that SCμM must be programmed via optical programmer and not three-wire-bus for this optical receiver code to function properly without additional modifications. Programming via three-wire bus disables the optical receiver by default.

The RF timer module is used by SCμM to measure the pulse width and time of the lighthouse signals. This timer's speed is based on HCLOCK, so increasing HCLOCK's frequency will increase the timing resolution of the lighthouse firmware. The code snippet below will set the HCLOCK divider to 2, which should yield an HCLOCK frequency of 10 MHz. The RF timer divider passthrough function should bypass the divider so that HCLOCK is 20 MHz, but that function didn't seem to behave as it should, as this code produces an HCLOCK frequency of 10 MHz.

```
// Set HCLK divider to 2
clear_asc_bit(57);
clear_asc_bit(56);
clear_asc_bit(55);
clear_asc_bit(54);
clear_asc_bit(53);
set_asc_bit(52);//inverted
set_asc_bit(51);
clear_asc_bit(50);

// Set RF Timer divider to pass through so that RF Timer is 20 MHz
set_asc_bit(36);
```

It is also important to connect VDDIO to an appropriate voltage, like VBAT, in order for the GPIOs to function correctly.

SCμM Lighthouse Hardware Configuration Checklist

1. Electrically connect GPIO pins 1, 8, and 9 together
2. Decrease VDDAO voltage to improve optical receiver performance
3. Program via optical bootloader and not three-wire bus
4. Configure GPIO banks
5. Enable relevant interrupts and GPIO pins
6. Increase RF timer frequency
7. Connect VDDIO to VBAT

B.2 Interrupt Implementation

A callback function named **void lh_int_cb(int level)** is called by both the the active high interrupt and active low interrupt: the active high interrupt passes 1 as an argument, and the active low interrupt passes 0 as an argument. Note: this use of the same function for both interrupts could potentially cause issues if one of these interrupts is called while the other is still executing code within this function; this could be causing the infrequent crashing seen due to these interrupts. This function detects rising and falling edges in the optical receiver's signal and records their times using the RF timer. Figure B.2 shows the sequence of events that make up this implementation of interrupts for lighthouse pulse detection. Once a consecutive rising and falling edge are detected, the function can calculate the width of the pulse, which contains information on the pulse's type.

A simple debounce counter is used to debounce the pulse. A debounce counter is used to count the number of times the active high level-sensitive interrupt triggers. Once this threshold is reached, the the state of the interrupt handler will transition to high and the active high interrupt is disabled. This is effectively enforcing a minimum "on" time for the interrupt to be triggered. This is useful for filtering out very short pulses (under 10 μs) in the receiver signal, which are common.

B.3 Pulse Type Classification

The pulse width contains important information about the type of information the pulse is carrying. The pulse types are enumerated in Table B.1. The skip pulse is used to indicate that no sweep is coming for that specific lighthouse, more information on processing multiple lighthouse pulses can be found in Figure 4.6.

The pulses are classified based on pulse width, which in the following code is measured in 10 MHz timer ticks.

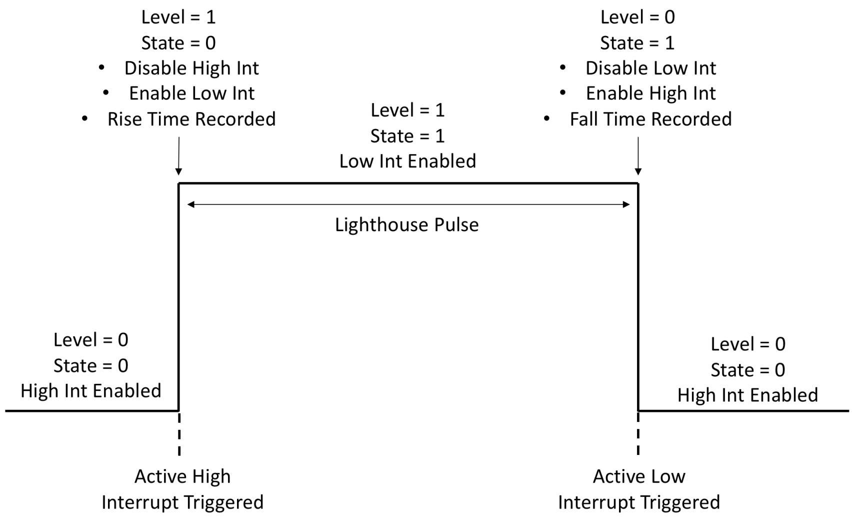


Figure B.2: Illustration of lighthouse pulse detection using level-sensitive interrupts.

Pulse Type	Description
T_A	Sync pulse announcing azimuth sweep
T_E	Sync pulse announcing elevation sweep
T_S	Sync pulse announcing skip (no sweep)
T_{sweep}	Full 180 degree sweep of the laser

Table B.1: Duration of the pulse activity of an HTC Vive Lighthouse.

```
pulse_width = (timestamp_fall - timestamp_rise);

// Identify what kind of pulse this was
pulse_type = INVALID;

if(pulse_width < 585  && pulse_width > 100 ){
    pulse_type = LASER; // Laser sweep
}
else if(pulse_width < 675  && pulse_width > 585 ){
    pulse_type = AZ; // Azimuth sync, data=0, skip = 0
}
else if(pulse_width >= 675  && pulse_width < 781 ){
    pulse_type = EL; // Elevation sync, data=0, skip = 0
}
else if(pulse_width >= 781  && pulse_width < 885 ){
    pulse_type = AZ; // Azimuth sync, data=1, skip = 0
}
else if(pulse_width >= 885  && pulse_width < 989 ){
    pulse_type = EL; // Elevation sync, data=1, skip = 0
}
else if(pulse_width >= 989  && pulse_width < 1083 ){
    pulse_type = AZ_SKIP; //Azimuth sync, data=0, skip = 1
}
else if(pulse_width >= 1083  && pulse_width < 1200 ){
    pulse_type = EL_SKIP; //elevation sync, data=0, skip = 1
}
else if(pulse_width >= 1200  && pulse_width < 1300 ){
    pulse_type = AZ_SKIP; //Azimuth sync, data=1, skip = 1
}
else if(pulse_width >= 1300  && pulse_width < 1400 ){
    pulse_type = EL_SKIP; //Elevation sync, data=1, skip = 1
}else{
    pulse_type = INVALID;
}
```

B.4 Pulse Processing State Machine

Once the pulses are classified, they are passed as input to a state machine that processes the pulses. This state machine ensures that the pulses are valid before calculating a lighthouse measurement, measures the timings of the sync and laser pulses, and sends a packet containing a lighthouse measurement once a valid measurement is detected. Figure B.3 shows the state machine used for processing pulses from two lighthouses, which includes processing elevation sync, azimuth sync, skip, and laser sweep pulses. Skip pulses are used to indicate which lighthouse is actively sweeping and are only present when two lighthouse base stations are used and synchronized together. Fig. 4.6 in the next chapter provides an explanation of how two lighthouse stations synchronize their pulses in time.

B.5 Enabling QX Series Boards to Boot

A significant fraction of SCμM devices (approximately 50%) cannot cold boot and are not programmable. SCμM development boards that include these chips are usually denoted as a QX# board. Fortunately, it is possible to add hardware workarounds to get these devices to operate effectively. If the VDDD rail, on pin VDD_LDO_OUT, of these chips is externally increased, they will be able to bootload and successfully run programs. This is colloquially

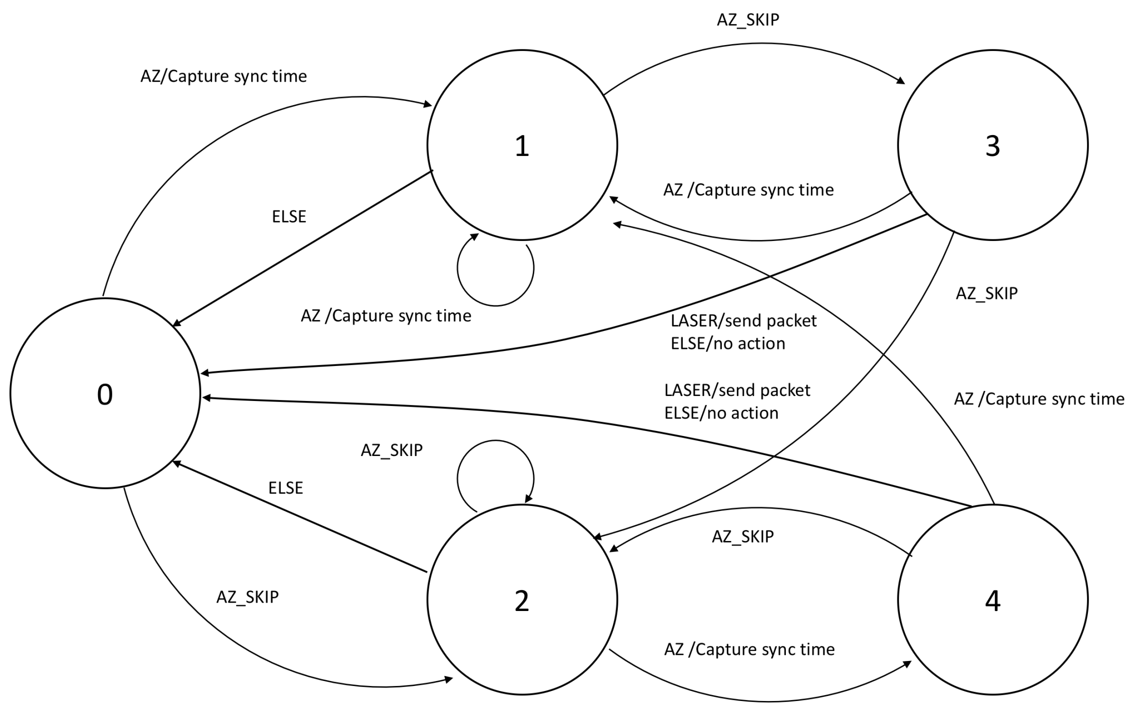


Figure B.3: State machine for processing lighthouse pulses containing azimuth measurements. The pulse processing for elevation measurements is identical to elevation, so its state machine is not shown. During state transitions that are triggered by receiving an azimuth sync pulse (AZ), the state machine records the time of the pulse's rising edge. During laser sweep pulse (LASER) transitions, the state machine will transmit a packet containing the time of the sync pulse's rising edge and the time of the laser pulse's rising edge, which can be used to calculate the SCμM's azimuth angle relative to the lighthouse base station. State transitions with "/*" execute the action "*" during the transition. State transitions without "/*" do not have any accompanying actions.

called the "VDDD tap". For example, the VDDD voltage on board QX3 must be forced to at least 0.93 V in order to program and boot correctly; note that this threshold voltage may vary over various chips. This workaround can be implemented with an external power supply or voltage regulator. A convenient solution is to add a 1.2 V LDO regulator to the SCμM development board, which can be powered by the VBAT of the board (Figure B.4). Sometimes it is possible to get these boards to bootload by only briefly raising the VDDD voltage, but it seems to be more reliable to use the LDO solution.

B.6 Optical Programming

Successfully programming SCμM with an optical programmer takes some trial and error and finesse. Programmer alignment and distance relative to the SCμM chip are the most

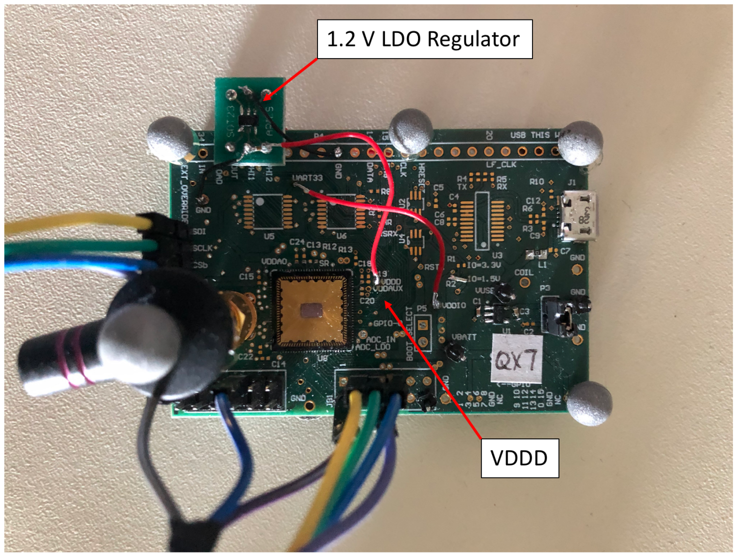


Figure B.4: A QX series board with LDO workaround. The 1.2 V LDO is powered by the board's VBAT rail and provides a 1.2 V output to SCμM's VDDD connection.

important variables for successfully programming. Usually, placing the programming LED 1-2 cm above SCμM's optical reciever provides reliable programming results. CAUTION: the infrared programming LED is not eyesafe while active; do not look directly into the LED ever and keep the LED at least 3 cm away from your eyes. The optical programming variables are also an important component to optical programming. The first number is the width of the high pulse in the "1" bit, the second is the width of the low pulse in the "1" bit, the third is the width of the high pulse in the "0" bit, and the fourth is the width of the low pulse in the "0" bit. The most important number is the third one. Section 7.4 of the SCμM user guide, found at `https://crystalfree.atlassian.net/`, provides more information about this. The ideal value will vary from chip to chip and can range from 2 to 10. The following bootloading code is an excerpt from a MATLAB optical programming script and contains settings that are known to work for board QX7.

```
% Configure parameters for optical TX
fprintf(s2,'configopt');
fprintf(s2,num2str(80));
fprintf(s2,num2str(80));
fprintf(s2,num2str(3)); %qx7 really likes 3
fprintf(s2,num2str(80));
```

B.7 Radio Calibration

Since SCμM doesn't use a crystal oscillator for keeping frequency, radio calibration is required to learn the correct local oscillator settings that correspond to IEEE802.15.4 channels. If the environmental operating conditions (e.g. temperature) of the target SCμM stay relatively static, this calibration process generally only needs to be done once for each chip. Whenever temperature, VBAT, or VDDD change, the frequency characteristics of the oscillator will change and the chip will need to be recalibrated. Other conditions that have been observed to change the oscillator frequency include connecting GPIOs together and holding the development board near the GPIOs or touching the antenna.

Multiple methods of calibration are possible [119, 25, 66]. The simplest, but most labor intensive, is to use trial and error to find oscillator settings that correspond to 802.15.4 channel frequencies. This more or less requires the use of a spectrum analyzer to see the output frequency of SCμM as it is sending packets. A better method of calibration utilizes a frequency sweep, where SCμM sweeps across oscillator codes and transmits a packet on each code. This packet contains the current oscillator code. An OpenMote [67] is programmed to listen for packets on a single channel and print out the contents of any packets it receives. When the OpenMote receives the a SCμM packet, it will print out the SCμM oscillator code that corresponds to its current channel. This code can then be hard-coded into the SCμM firmware, allowing it to transmit packets on the OpenMote's frequency. The code below shows this transmit sweep process, where the oscillator codes are placed into an IMU packet and transmitted.

```
// Enable the TX. NB: Requires 50us for frequency settling
// transient.
radio_txEnable();

while (1) {
    for (coarse=20; coarse<27; coarse++) {
        for (mid=0; mid<32; mid++) {
            for (fine=0; fine<32; fine++) {

                sweep_setting.coarse.value = coarse & 0x1F;
                sweep_setting.mid.value = mid & 0x1F;
                sweep_setting.fine.value = fine & 0x1F;

                // Construct the packet
                // with payload {coarse, mid, fine} in
                // separate bytes

                //place packet type code in first byte of packet
                send_packet[0] = IMU_CODE;
                //place coarse code into packet (lsb first)
                send_packet[1] = sweep_setting.coarse.bytes[0];
                send_packet[2] = sweep_setting.coarse.bytes[1];

                //place mid code into packet
                send_packet[3] = sweep_setting.mid.bytes[0];
                send_packet[4] = sweep_setting.mid.bytes[1];

                //place fine code into packet
                send_packet[5] = sweep_setting.fine.bytes[0];
                send_packet[6] = sweep_setting.fine.bytes[1];

                //send_packet is a global variable that
                //radio_loadPacket(int length) can access

                //load packet
                radio_loadPacket(7);


                // Set the LC frequency
                LC_FREQCHANGE(coarse&0x1F, mid&0x1F, fine&0x1F);

                //gpo toggle on all gpos
                GPIO_REG__OUTPUT = ~GPIO_REG__OUTPUT;

                // TODO: Wait for at least 50us
                for (i=0; i<5000; i++) {}

                //send packet
                radio_txNow();

                // Send packets thrice for redundancy
                for(i=0; i<iterations; i++) {
                    for (i=0; i<5000; i++) {}
                    //send packet
                    radio_txNow();
                }
            }
        }
    }
}
```

Further improvements can be made on this calibration method to increase ease of use. For example, Chang et al. [120] demonstrates a "QuickCal" algorithm for automatically calibrating the channel frequencies of SCμM so the mote can successfully join a 6TiSCH network, which would also be applicable in a SCμM lighthouse application. A limited version of this automatic calibration algorithm could also be implemented.

Note: code for SCμM lighthouse implementation can be found at `https://github.com/PisterLab/scum-test-code`.

Appendix C

Lighthouse RF Synchronization

Currently, the lighthouse localization system utilizes an omnidirectional infrared pulse to inform nodes of the phase of the laser sweep. Nodes can then use the time difference between the laser sweep pulse and the synchronization pulse to determine their relative heading to the lighthouse base station. The downside of this approach is that the synchronization pulse limits the range of the lighthouse, where the laser sweep has more range than the omnidirectional pulse (Figure C.1).

One solution to this problem is replacing the optical pulse with an RF synchronization signal. The range of an RF signal would be larger than the optical synchronization pulse, making the laser sweeps the limiting factor on range.

A natural wireless communication for low-power, miniature motes used in lighthouse localization is OpenWSN, a mesh networking wireless stack intended for wireless sensor networks of nodes with low-power and lossy communications [42]. OpenWSN utilizes time-synchronized channel-hopping [108] to decrease the radio duty cycle of the sensor nodes in order to achieve low overall power consumption. A scheduling matrix (Figure C.2) is used to determine which frequency and which time nodes within the network are communicating and listening on. This synchronization allows nodes to turn off their radio during times in the schedule that they know there won't be another node communicating with them.

OpenWSN relies on accurate time synchronization for it's scheduling scheme to function. Fortunately, this time synchronization can also be used to efficiently replace the lighthouse

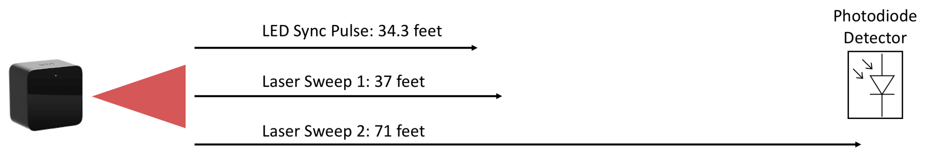


Figure C.1: Range comparison between laser sweeps and omnidirectional synchronization pulses. The range was determined by increasing the distance between the photodiode receiver and the lighthouse beacon until the the light pulses could no longer be seen by the receiver.

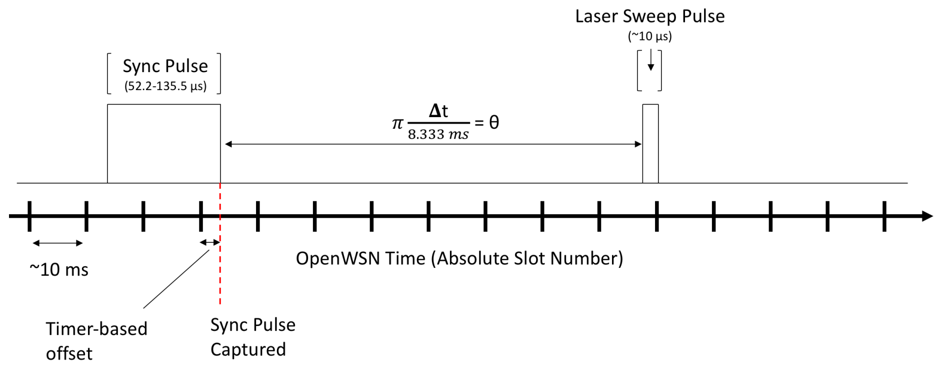


Figure C.2: Timing of Lighthouse OpenWSN Integration. OpenWSN uses the protocol 6TiSCH protocol [121] to implement time-synchronized channel hopping. In this case, the minimal 6TiSCH cell is the minimal required active slot in the schedule, and determines when nodes in the network can communicate with each other.

synchronization pulse with an RF synchronization scheme. A network-wide synchronized time base is used in OpenWSN to track network events and denote the current timeslot of the network. The Absolute Slot Number (ASN) is the current timeslot of the network and is usually a unit of 10 ms, depending on network settings. Finer timing with the ASN can be achieved by measuring the time offset within the ASN using a node's onboard timer module.

This **ASN + timer offset** paradigm can be used to capture the synchronized network time of a lighthouse omnidirectional pulse. A single network mote, in close range to the lighthouse beacon, can be used to detect the lighthouse sync pulse timing and broadcast that timing to the rest of the nodes in the network that may be out of range of the sync pulse. Instead of relaying the sync pulse everytime it occurs, this mote can leverage the pulse's periodic nature and provide the timing offset and clock drift of the 120 Hz sync pulse. The detecting mote will calculate the period of the sync pulse with respect to OpenWSN time and time of the pulse in ASN + offset time (Figure C.4). This information is periodically broadcast to the entire network via an Enhanced Beacon (EB) which includes an Information Element (IE), or small data container, that contains the period and current sync pulse timing (Figure C.3). This allows motes in the network to estimate when the synchronization pulses happen based on their period and offset, without requiring the sync mote to broadcast the time of every single sync pulse (Figure C.5). This avoids potential timing errors caused by latency in the network or missed packets. After a period of time, the sync mote can rebroadcast the period and timing information to counteract desynchronization. Localizing motes in an OpenWSN network can then compare the current ASN + offset time of their laser sweep pulse to their estimated sync pulse time in order to calculate their relative bearing to the lighthouse base station.

The RF lighthouse synchronization system was implemented using MIMSYs [40] equipped with photodiodes and running the OpenWSN stack. The synchronization MIMSY would

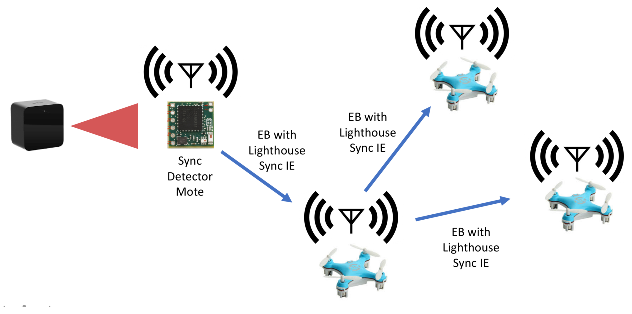


Figure C.3: Broadcast of Enhanced Beacons (EBs) containing Information Elements (IEs) with period and timing information of lighthouse sync pulses.

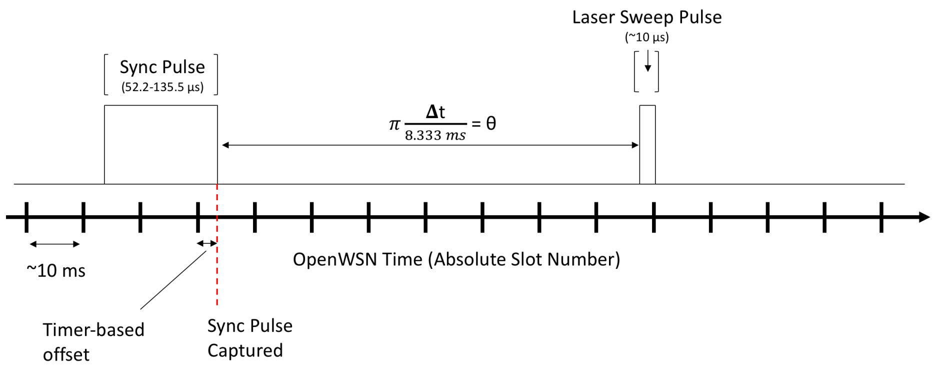


Figure C.4: Timing of Lighthouse OpenWSN Integration. The sync pulse time of arrival is captured in the network-synchronized time base, which is the time of the current ASN plus the timer measured offset between the the time the pulse occurred and the time corresponding to the beginning of the current timeslot.

detect the lighthouse sync pulse and calculate its period and record the time it occurred. The MIMSY would use the 32.768 kHz timer linked to its 32 kHz crystal oscillator to calculate the ASN offset of the sync pulse, which yields a timing resolution of 31.25 μs. Figure C.6 shows the drift error of the projected sync pulse measurement if the lighthouse period was assumed to be exactly 120 Hz, which underscores the importance of calculating the sync pulse period with respect with to OpenWSN time. When the period is calculated by the sync mote, this drift can be eliminated (Figure C.7). The mote periodically broadcasts this information to other motes in the system. An EB containing the most recent lighthouse sync pulse time is broadcast every 60 seconds. The error statistics for the project pulses vs. measured pulses are shown in Figure C.8. The 95 μs standard deviation of error yields a 0.072 radian error in heading, which translates to 7 cm of localization error at a range of 1 m. While this synchronization method does reduce the accuracy of lighthouse localization, it does demonstrate the feasibility of synchronizing the lighthouse sync pulse with an OpenWSN network. Further work can focus on using a faster timer to calculate the sync pulse timing

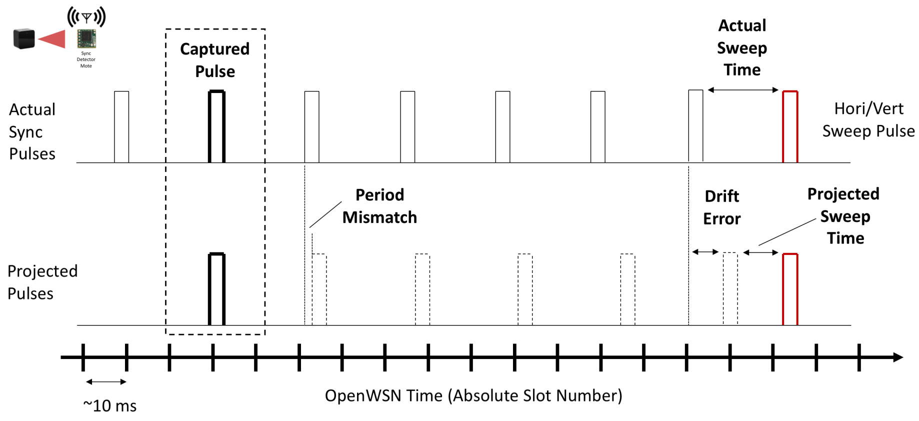


Figure C.5: Illustration of lighthouse sync pulse drift. Nodes in the network periodically receive the exact timing of synchronization pulses from the sync pulse detection node. Between these resynchronization packets, motes predict when the next sync pulses occur based on their period. The motes can then use these predicted pulses to localize when they receive laser sweeps. As the amount of time since the last sync pulse reference packet was sent increases, the projected pulse times will drift and accumulate error.

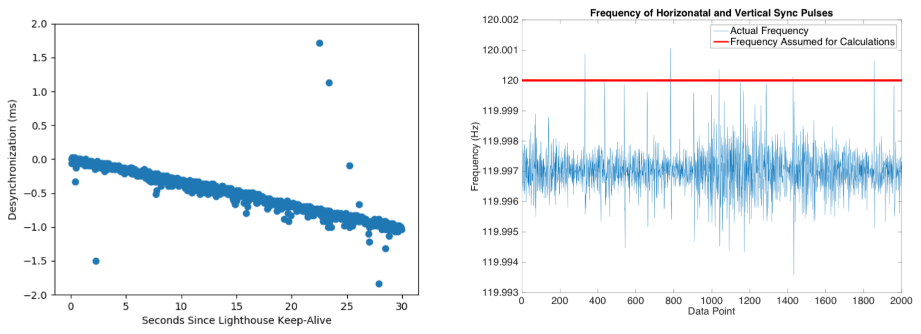


Figure C.6: Left: Illustration of drift between sync pulse projection and actual sync pulse. Right: Measured frequency of lighthouse sync pulses.

and period, which would improve the resolution and presumably the timing error.

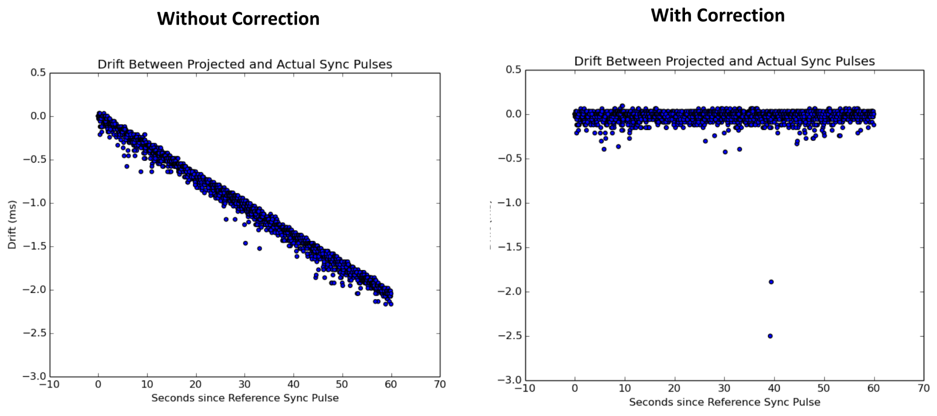


Figure C.7: Drift error of lighthouse sync pulses timing, with and without drift correction.

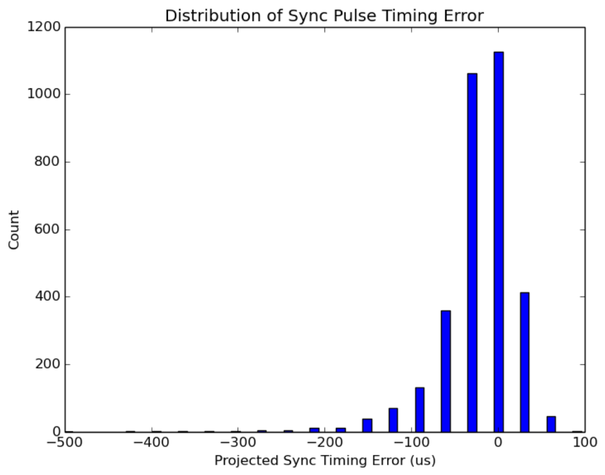


Figure C.8: Timing error of lighthouse synchronization pulse with respect to the projected sync pulse timings. The mean error was 32 μs, the standard deviation was 95 μs, and the timing resolution was 30.52 μs.

Bibliography

[1] Noah T Jafferis, E Farrell Helbling, Michael Karpelson, and Robert J Wood. "Untethered flight of an insect-sized flapping-wing microscale aerial vehicle". In: *Nature* 570.7762 (2019), pp. 491–495. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1322-0. URL: https://doi.org/10.1038/s41586-019-1322-0.

[2] Kevin Y. Ma, Pakpong Chirarattananon, Sawyer B. Fuller, and Robert J. Wood. "Controlled Flight of a Biologically Inspired, Insect-Scale Robot". In: *Science* 340.6132 (2013), pp. 603–607. ISSN: 0036-8075. DOI: 10.1126/science.1231806. eprint: https://science.sciencemag.org/content/340/6132/603.full.pdf. URL: https://science.sciencemag.org/content/340/6132/603.

[3] Xiaojun Yan, Mingjing Qi, and Liwei Lin. "Self-lifting artificial insect wings via electrostatic flapping actuators". In: *2015 28th IEEE International Conference on Micro Electro Mechanical Systems (MEMS)*. IEEE. 2015, pp. 22–25.

[4] Z. Liu, X. Yan, M. Qi, Y. Yang, X. Zhang, and L. Lin. "Lateral moving of an artificial flapping-wing insect driven by low voltage electromagnetic actuator". In: *2017 IEEE 30th International Conference on Micro Electro Mechanical Systems (MEMS)*. 2017, pp. 777–780.

[5] D. S. Drew, N. O. Lambert, C. B. Schindler, and K. S. J. Pister. "Toward Controlled Flight of the Ionocraft: A Flying Microrobot Using Electrohydrodynamic Thrust With Onboard Sensing and No Moving Parts". In: *IEEE Robotics and Automation Letters* 3.4 (Oct. 2018), pp. 2807–2813. ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2844461.

[6] Daniel S. Contreras, Daniel Drew, and Kristofer S. J. Pister. "First steps of a millimeter-scale walking silicon robot". In: *Solid State Sensors, Actuators, and Microsystems*. in press. 2017.

[7] Daniel S Contreras and Kristofer SJ Pister. "A six-legged MEMS silicon robot using multichip assembly". In: *Solid State Sensor, Actuators and Microsystems Workshop (Hilton Head)*. 2018.

[8] Ken Saito, Minami Takato, Yoshifumi Sekine, and Fumio Uchikoba. "Biomimetics Micro Robot with Active Hardware Neural Networks Locomotion Control and Insect-Like Switching Behaviour". In: *International Journal of Advanced Robotic Systems* 9.5

(2012), p. 226. DOI: `10.5772/54129`. eprint: `https://doi.org/10.5772/54129`. URL: `https://doi.org/10.5772/54129`.

[9] S. Hollar, A. Flynn, C. Bellew, and K.S.J. Pister. “Solar powered 10 mg silicon robot”. In: *The Sixteenth Annual International Conference on Micro Electro Mechanical Systems, 2003. MEMS-03 Kyoto. IEEE* (2003), pp. 706–711. ISSN: 1084-6999. DOI: `10.1109/MEMSYS.2003.1189847`.

[10] D. Vogtmann, R. S. Pierre, and S. Bergbreiter. “A 25 MG magnetically actuated microrobot walking at $>$ 5 body lengths/sec”. In: *2017 IEEE 30th International Conference on Micro Electro Mechanical Systems (MEMS)*. 2017, pp. 179–182.

[11] S. Bergbreiter and K. S. J. Pister. “Design of an Autonomous Jumping Microrobot”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 447–453.

[12] W. A. Churaman, A. P. Gerratt, and S. Bergbreiter. “First leaps toward jumping microrobots”. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 1680–1686.

[13] C. B. Schindler, J. T. Greenspun, H. C. Gomez, and K. S. J. Pister. “A Jumping Silicon Microrobot with Electrostatic Inchworm Motors and Energy Storing Substrate Springs”. In: *2019 20th International Conference on Solid-State Sensors, Actuators and Microsystems Eurosensors XXXIII (TRANSDUCERS EUROSENSORS XXXIII)*. 2019, pp. 88–91.

[14] Joseph Greenspun and Kristofer SJ Pister. “First leaps of an electrostatic inchworm motor-driven jumping microrobot”. In: *Micro Electro Mechanical Systems (MEMS)*. 2018.

[15] M. Noh, S. Kim, S. An, J. Koh, and K. Cho. “Flea-Inspired Catapult Mechanism for Miniature Jumping Robots”. In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1007–1018.

[16] Palak Bhushan and Claire Tomlin. *An Insect-scale Untethered Laser-powered Jumping Microrobot*. 2019. arXiv: `1908.03282 [cs.RO]`.

[17] RJ Wood, E Steltz, and RS Fearing. “Optimal energy density piezoelectric bending actuators”. In: *Sensors and Actuators A: Physical* 119.2 (2005), pp. 476–488.

[18] Robert J Wood, Srinath Avadhanula, Erik Steltz, M Seeman, Jon Entwistle, Abraham Bachrach, G Barrows, Seth Sanders, and Ronald S Fearing. “An autonomous palm-sized gliding micro air vehicle”. In: *IEEE Robotics & Automation Magazine* 14.2 (2007), pp. 82–91.

[19] E. Y. Erdem, Y. Chen, M. Mohebbi, J. W. Suh, G. T. A. Kovacs, R. B. Darling, and K. F. Böhringer. “Thermally Actuated Omnidirectional Walking Microrobot”. In: *Journal of Microelectromechanical Systems* 19.3 (2010), pp. 433–442.

[20] Carter S Haines, Márcio D Lima, Na Li, Geoffrey M Spinks, Javad Foroughi, John DW Madden, Shi Hyeong Kim, Shaoli Fang, Mônica Jung De Andrade, Fatma Göktepe, et al. "Artificial muscles from fishing line and sewing thread". In: *science* 343.6173 (2014), pp. 868–872.

[21] I Penskiy and S Bergbreiter. "Optimized electrostatic inchworm motors". In: *J. Micromech. Microeng.* 23 (2013). DOI: `10.1088/0960-1317/23/1/015018`.

[22] Richard Yeh, Seth Hollar, and Kristofer SJ Pister. "Single mask, large force, and large displacement electrostatic linear inchworm motors". In: *Journal of Microelectromechanical Systems* 11.4 (2002), pp. 330–336.

[23] A. Weinstein, A. Cho, G. Loianno, and V. Kumar. "Visual Inertial Odometry Swarm: An Autonomous Swarm of Vision-Based Quadrotors". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1801–1807.

[24] G. Loianno, C. Brunner, G. McGrath, and V. Kumar. "Estimation, Control, and Planning for Aggressive Flight With a Small Quadrotor With a Single Camera and IMU". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 404–411.

[25] Filip Maksimovic, Brad Wheeler, David C Burnett, Osama Khan, Sahar Mesri, Ioana Suciu, Lydia Lee, Alex Moreno, Arvind Sundararajan, Bob Zhou, et al. "A Crystal-Free Single-Chip Micro Mote with Integrated 802.15. 4 Compatible Transceiver, sub-mW BLE Compatible Beacon Transmitter, and Cortex M0". In: *2019 Symposium on VLSI Circuits*. IEEE. 2019, pp. C88–C89.

[26] Y. Lee, S. Bang, I. Lee, Y. Kim, G. Kim, M. H. Ghaed, P. Pannuto, P. Dutta, D. Sylvester, and D. Blaauw. "A Modular 1 mm^3 Die-Stacked Sensing Platform With Low Power I^{2}C Inter-Die Communication and Multi-Modal Energy Harvesting". In: *IEEE Journal of Solid-State Circuits* 48.1 (Jan. 2013), pp. 229–243. ISSN: 0018-9200. DOI: `10.1109/JSSC.2012.2221233`.

[27] Nikon Metrology. *iSpace: Large Volume Metrology, tracking, and positioning*. Brochure. 2010. URL: `https://www.nikonmetrology.com/en-gb/product/igps`.

[28] HTC Vive. *Vive VR System*. Webpage. 2019. URL: `https://www.vive.com/us/product/vive-virtual-reality-system/`.

[29] David Hambling. *Swarm Troopers: How small drones will conquer the world*. Archangel Ink, 2015.

[30] Neal Stephenson. *The Diamond Age*. Bantam, 1995.

[31] W. Lindsay, D. Teasdale, V. Milanovic, K. Pister, and C. Fernandez-Pello. "Thrust and electrical power from solid propellant microrockets". In: *MEMS*. 2001.

[32] Dana Teasdale. "Solid propellant microrockets". In: *Master Thesis of University California at Berkeley* (2000).

[33] Mirko Kovac, Maria Bendana, Rohit Krishnan, Jessica Burton, Micheal Smith, and Robert J Wood. "Multi-stage micro rockets for robotic insects". In: *Robotics: Science and Systems VIII* (2013), p. 185.

[34] M. Nieuwenhuisen, J. Quenzel, M. Beul, D. Droeschel, S. Houben, and S. Behnke. "ChimneySpector: Autonomous MAV-based indoor chimney inspection employing 3D laser localization and textured surface reconstruction". In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. June 2017, pp. 278–285. DOI: `10.1109/ICUAS.2017.7991427`.

[35] D. Orol, J. Das, L. Vacek, I. Orr, M. Paret, C. J. Taylor, and V. Kumar. "An aerial phytobiopsy system: Design, evaluation, and lessons learned". In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. June 2017, pp. 188–195. DOI: `10.1109/ICUAS.2017.7991431`.

[36] D. S. Drew, B. Kilberg, and K. S. J. Pister. "Future mesh-networked pico air vehicles". In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. June 2017, pp. 1075–1082. DOI: `10.1109/ICUAS.2017.7991503`.

[37] Brian G Kilberg, Daniel S Contreras, Joseph Greenspun, and Kristofer SJ Pister. "MEMS aerodynamic control surfaces for millimeter-scale rockets". In: *2017 International Conference on Manipulation, Automation and Robotics at Small Scales (MARSS)*. IEEE. 2017, pp. 1–5.

[38] Brian Kilberg, Daniel Contreras, Joseph Greenspun, Hani Gomez, Eric Liu, and Kristofer S. J. Pister. "Aerodynamic Forces Generated by an Electrostatic Inchworm Motor-Actuated MEMS Control Surface Integrated on a Force-Sensing Platform". In: *Solid State Sensor, Actuators and Microsystems Workshop (Hilton Head)*. Transducers Research Foundation. 2018, pp. 306–309. DOI: `10.31438/trf.hh2018.87`.

[39] A. M. Rauf, B. G. Kilberg, C. B. Schindler, S. A. Park, and K. S. J. Pister. "Towards Aerodynamic Control of Miniature Rockets with MEMS Control Surfaces". In: *2020 IEEE 33rd International Conference on Micro Electro Mechanical Systems (MEMS)*. 2020, pp. 523–526.

[40] Craig B Schindler, Daniel S Drew, Brian G Kilberg, Felipe MR Campos, Soichiro Yanase, and Kristofer SJ Pister. "MIMSY: The Micro Inertial Measurement System for the Internet of Things". In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE. 2019, pp. 329–334.

[41] Alex Moreno, Filip Maksimovic, Lydia Lee, Brian Kilberg, Craig Schindler, Hani Gomez, Daniel Teal, Dillon Acker-James, Andrew Fearing, Jan S. Rentmeister, Jason T. Stauth, and Kristofer S. J. Pister. "Single-Chip micro-Mote for Microrobotic Platforms". In: *GOMACtech*. 2020.

[42] Thomas Watteyne, Xavier Vilajosana, Branko Kerkez, Fabien Chraim, Kevin Weekly, Qin Wang, Steven Glaser, and Kris Pister. "OpenWSN: a standards-based low-power wireless development environment". In: *Transactions on Emerging Telecommunications Technologies* 23.5 (2012), pp. 480–493.

[43] Chih-Ming Ho and Yu-Chong Tai. "Micro-electro-mechanical-systems (MEMS) and fluid flows". In: *Annual Review of Fluid Mechanics* 30.1 (1998), pp. 579–612.

[44] A Linga Murthy and Y Krishna. "Design and Analysis of MEMS-based Microballoon Actuators for Aerodynamic Control of Flight Vehicles". In: *Defence Science Journal* 59.6 (2009), p. 642.

[45] Sergey Edward Lyshevski. "Smart flight control surfaces with microelectromechanical systems". In: *IEEE Transactions on Aerospace and Electronic Systems* 38.2 (2002), pp. 543–552.

[46] R. Low, M. Deeds, D.L. Jean, and C. Hovland. *MEMS control surface for projectile steering*. US Patent 7,628,352. Dec. 2009. URL: `https://www.google.com/patents/US7628352`.

[47] Adam Huang, Chris Folk, Chih-Ming Ho, Z Liu, Wesley W Chu, Yong Xu, and Yu-Chong Tai. "Gryphon M 3 system: integration of MEMS for flight control". In: *MEMS Components and Applications for Industry, Automobiles, Aerospace, and Communication*. Vol. 4559. International Society for Optics and Photonics. 2001, pp. 85–95.

[48] C Folk and C-M Ho. "Micro-actuators for control of delta wing with sharp leading edge". In: *39th Aerospace Sciences Meeting and Exhibit*. 2001, p. 121.

[49] Adam Huang, Chris Folk, Chris Silva, Brian Christensen, Yih-Far Chen, Chih-Ming Ho, Fukang Jiang, Charles Grosjean, and Yu-Chong Tai. "Applications of MEMS devices to delta wing aircraft-From concept development to transonic flight test". In: *39th Aerospace Sciences Meeting and Exhibit*. 2001, p. 124.

[50] Robert J Wood, Ben Finio, Michael Karpelson, Kevin Ma, Néstor Osvaldo Pérez-Arancibia, Pratheev S Sreetharan, Hiroto Tanaka, and John Peter Whitney. "Progress on 'pico'air vehicles". In: *The International Journal of Robotics Research* 31.11 (2012), pp. 1292–1302.

[51] Daniel S Drew, Brian Kilberg, and Kristofer SJ Pister. "Future mesh-networked pico air vehicles". In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2017, pp. 1075–1082.

[52] Daniel S Contreras and Kristofer S J Pister. "Durability of Silicon Pin-Joints for Microrobotics". In: *MARSS*. 2016.

[53] Daniel Contreras. "Walking Silicon: Actuators and Legs for Small-Scale Terrestrial Robots". PhD thesis. EECS Department, University of California, Berkeley, May 2019. URL: `http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-18.html`.

[54] Jan S. Rentmeister, Kristofer S. J. Pister, and Jason T. Stauth. "A 120-330 V, sub-μA, optically powered microrobotic drive IC for DARPA SHRIMP," in: *GOMACtech*. 2020.

[55] Gilford Norman Ward. *Linearized theory of steady high-speed flow*. Cambridge University Press, 2016.

[56] C. B. Schindler, H. C. Gomez, D. Acker-James, D. Teal, W. Li, and K. S. J. Pister. "15 Millinewton Force, 1 Millimeter Displacement, Low-Power MEMS Gripper". In: *2020 IEEE 33rd International Conference on Micro Electro Mechanical Systems (MEMS)*. 2020, pp. 485–488.

[57] Brian Yang, Grant Wang, Roberto Calandra, Daniel Contreras, Sergey Levine, and Kristofer Pister. "Learning flexible and reusable locomotion primitives for a microrobot". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1904–1911.

[58] P Estevez, JM Bank, M Porta, J Wei, PM Sarro, M Tichem, and U Staufer. "6 DOF force and torque sensor for micro-manipulation applications". In: *Sensors and Actuators A: Physical* 186 (2012), pp. 86–93.

[59] Felix Beyeler, Simon Muntwyler, and Bradley J Nelson. "Design and calibration of a microfabricated 6-axis force-torque sensor for microrobotic applications". In: *2009 IEEE international conference on robotics and automation*. IEEE. 2009, pp. 520–525.

[60] Wei Zhang, Kim Boon Lua, A Senthil Kumar, Tee Tai Lim, Khoon Seng Yeo, Guangya Zhou, et al. "Design and characterization of a silicon piezoresistive three-axial force sensor for micro-flapping wing MAV applications". In: *International Conference on Experimental Mechanics 2014*. Vol. 9302. International Society for Optics and Photonics. 2015, 93023E.

[61] Qingsong Xu. "Design, fabrication, and testing of an MEMS microgripper with dual-axis force sensor". In: *IEEE Sensors Journal* 15.10 (2015), pp. 6017–6026.

[62] Joseph M Kahn, Randy H Katz, and Kristofer SJ Pister. "Next century challenges: mobile networking for "Smart Dust"". In: *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. 1999, pp. 271–278.

[63] Kay Römer. "The lighthouse location system for smart dust". In: *Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM. 2003, pp. 15–30.

[64] Miguel Borges, Andrew Symington, Brian Coltin, Trey Smith, and Rodrigo Ventura. "HTC Vive: Analysis and accuracy improvement". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 2610–2615.

[65] Brad Wheeler, Andrew Ng, Brian Kilberg, Filip Maksimovic, and Kristofer S. J. Pister. "A Low-Power Optical Receiver for Contact-free Programming and 3D Localization of Autonomous Microsystems". In: *IEEE UEMCON*. 2019.

[66] Tengfei Chang, Thomas Watteyne, Brad Wheeler, Filip Maksimovic, Osama Khan, Sahar Mesri, Lydia Lee, Ioana Suciu, David Burnett, Xavier Vilajosana, et al. "6TiSCH on SCμM: Running a Synchronized Protocol Stack without Crystals". In: *Sensors* 20.7 (2020), p. 1912.

[67] Xavier Vilajosana, Pere Tuset, Thomas Watteyne, and Kris Pister. "OpenMote: open-source prototyping platform for the industrial IoT". In: *International Conference on Ad Hoc Networks*. Springer. 2015, pp. 211–222.

[68] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2003.

[69] Gordon Wetzstein. *EE 267 Course Notes: 6-DOF Pose Tracking with the VRDuino*. Stanford University, Mar. 2019.

[70] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. "EPnP: An Accurate O(n) Solution to the PnP Problem". In: *International Journal of Computer Vision* 81 (Feb. 2009).

[71] Zhengyou Zhang. "A Flexible New Technique for Camera Calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (Nov. 2000), pp. 1330–1334.

[72] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[73] Guobin Chang. "Robust Kalman Filtering Based on Mahalanobis Distance as Outlier Judging Criterion". In: *Journal of Geodesy* 88.4 (2014), pp. 391–401.

[74] Mohinder S Grewal and Angus P Andrews. *Kalman filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2014.

[75] Felipe M. R. Campos, Craig Schindler, Brian Kilberg, and Kristofer S. J. Pister. "Lighthouse Localization of Wireless Sensor Networks for Latency-Bounded, High-Reliability Industrial Automation Tasks". In: *International Conference on Factory Communication Systems (WFCS)*. IEEE. 2020.

[76] PC Mahalanobis. "On the generalised distance in statistics". In: *in Proceedings National Institute of Science, India* 2.1 (1936), pp. 49–55.

[77] Rudolph Emil Kalman. "A new approach to linear filtering and prediction problems". In: (1960).

[78] Eric A Wan and Rudolph Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee. 2000, pp. 153–158.

[79] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. "On sequential Monte Carlo sampling methods for Bayesian filtering". In: *Statistics and computing* 10.3 (2000), pp. 197–208.

[80] F Landis Markley. "Attitude error representations for Kalman filtering". In: *Journal of guidance, control, and dynamics* 26.2 (2003), pp. 311–317.

[81] I. y. Bar-Itzhack and Y. Oshman. "Attitude Determination from Vector Observations: Quaternion Estimation". In: *IEEE Transactions on Aerospace and Electronic Systems* AES-21.1 (1985), pp. 128–136.

[82] Mark W Mueller, Markus Hehn, and Raffaello D'Andrea. "Covariance correction step for kalman filtering with an attitude". In: *Journal of Guidance, Control, and Dynamics* 40.9 (2016), pp. 2301–2306.

[83] M. W. Mueller, M. Hamer, and R. D'Andrea. "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 1730–1736.

[84] M. Greiff, A. Robertsson, and K. Berntorp. "Performance Bounds in Positioning with the VIVE Lighthouse System". In: *2019 22th International Conference on Information Fusion (FUSION)*. 2019, pp. 1–8.

[85] Sebastian Madgwick. "An efficient orientation filter for inertial and inertial/magnetic sensor arrays". In: *Report x-io and University of Bristol (UK)* 25 (2010), pp. 113–118.

[86] Pradipta Ghosh, Andrea Gasparri, Jiong Jin, and Bhaskar Krishnamachari. "Robotic Wireless Sensor Networks". In: *arXiv preprint arXiv:1705.05415* (2017).

[87] Jacques Penders, Lyuba Alboul, Ulf Witkowski, Amir Naghsh, Joan Saez-Pons, Stefan Herbrechtsmeier, and Mohamed El-Habbal. "A robot swarm assisting a human firefighter". In: *Advanced Robotics* 25.1-2 (2011), pp. 93–117.

[88] Maxim A Batalin and Gaurav S Sukhatme. "Coverage, exploration and deployment by a mobile robot and communication network". In: *Telecommunication Systems* 26.2-4 (2004), pp. 181–196.

[89] Nikolaus Correll, Jonathan Bachrach, Daniel Vickery, and Daniela Rus. "Ad-hoc wireless network coverage with networked robots that cannot localize". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 3878–3885.

[90] M. D. Weiss, J. Peak, and T. Schwengler. "A Statistical Radio Range Model for a Robot MANET in a Subterranean Mine". In: *IEEE Transactions on Vehicular Technology* 57.5 (Sept. 2008), pp. 2658–2666. ISSN: 0018-9545. DOI: `10.1109/TVT.2007.912606`.

[91] Nick Iliev and Igor Paprotny. "Review and comparison of spatial localization methods for low-power wireless sensor networks". In: *IEEE Sensors Journal* 15.10 (2015), pp. 5971–5987.

[92] Guoqiang Mao, Barış Fidan, and Brian DO Anderson. "Wireless sensor network localization techniques". In: *Computer networks* 51.10 (2007), pp. 2529–2553.

[93] Pierre Merriaux, Yohan Dupuis, Rémi Boutteau, Pascal Vasseur, and Xavier Savatier. "A study of vicon system positioning performance". In: *Sensors* 17.7 (2017), p. 1591.

[94] H. Durrant-Whyte and T. Bailey. “Simultaneous localization and mapping: part I”. In: *IEEE Robotics Automation Magazine* 13.2 (June 2006), pp. 99–110. ISSN: 1070-9932. DOI: `10.1109/MRA.2006.1638022`.

[95] Brett Warneke, Matt Last, Brian Liebowitz, and Kristofer SJ Pister. “Smart dust: Communicating with a cubic-millimeter computer”. In: *Computer* 34.1 (2001), pp. 44–51.

[96] Yi Yang, Dongdong Weng, Dong Li, and Hang Xun. “An Improved Method of Pose Estimation for Lighthouse Base Station Extension”. In: *Sensors* 17.10 (2017), p. 2411.

[97] Brian E Nemsick, Austin D Buchan, Anusha Nagabandi, Ronald S Fearing, and Avideh Zakhor. “Cooperative inchworm localization with a low cost team”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 6323–6330.

[98] Robert Grabowski, Luis E Navarro-Serment, Christiaan JJ Paredis, and Pradeep K Khosla. “Heterogeneous teams of modular robots for mapping and exploration”. In: *Autonomous Robots* 8.3 (2000), pp. 293–308.

[99] Stergios I Roumeliotis and George A Bekey. “Distributed multirobot localization”. In: *IEEE Transactions on Robotics and Automation* 18.5 (2002), pp. 781–795.

[100] F. R. Fabresse, F. Caballero, and A. Ollero. “Decentralized simultaneous localization and mapping for multiple aerial vehicles using range-only sensors”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. May 2015, pp. 6408–6414. DOI: `10.1109/ICRA.2015.7140099`.

[101] T. R. Wanasinghe, G. K. I. Mann, and R. G. Gosine. “Distributed collaborative localization for a heterogeneous multi-robot system”. In: *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*. May 2014, pp. 1–6. DOI: `10.1109/CCECE.2014.6900998`.

[102] G. M. Hoffmann and C. J. Tomlin. “Mobile Sensor Network Control Using Mutual Information Methods and Particle Filters”. In: *IEEE Transactions on Automatic Control* 55.1 (Jan. 2010), pp. 32–47. DOI: `10.1109/TAC.2009.2034206`.

[103] S.E. Hammel, P.T. Liu, E.J. Hilliard, and K.F. Gong. “Optimal observer motion for localization with bearing measurements”. In: *Computers & Mathematics with Applications* 18.1 (1989), pp. 171–180. ISSN: 0898-1221. DOI: `https://doi.org/10.1016/0898-1221(89)90134-X`. URL: `http://www.sciencedirect.com/science/article/pii/089812218990134X`.

[104] B. Grocholsky, A. Makarenko, and H. Durrant-Whyte. “Information-theoretic coordinated control of multiple sensor platforms”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 1. Sept. 2003, 1521–1526 vol.1. DOI: `10.1109/ROBOT.2003.1241807`.

[105] Sarah Bergbreiter, Ankur Mehta, Kristofer SJ Pister, et al. "PhotoBeacon: design of an optical system for localization and communication in multi-robot systems." In: *ROBOCOMM*. 2007, p. 5.

[106] James F Roberts, Timothy Stirling, Jean-Christophe Zufferey, and Dario Floreano. "3-D relative positioning sensor for indoor flying robots". In: *Autonomous Robots* 33.1-2 (2012), pp. 5–20.

[107] T. Stirling, J. Roberts, J. Zufferey, and D. Floreano. "Indoor navigation with a swarm of flying robots". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 4641–4647.

[108] Kris Pister and Lance Doherty. "TSMP: Time synchronized mesh protocol". In: *IASTED Distributed Sensor Networks* (2008), pp. 391–398.

[109] Thomas Watteyne, Ankur Mehta, and Kris Pister. "Reliability through frequency diversity: why channel hopping makes sense". In: *Proceedings of the 6th ACM symposium on Performance Evaluation of Wireless Ad hoc, Sensor, and Ubiquitous Networks*. ACM. 2009, pp. 116–123.

[110] *IEEE 802.15.4e-2012 - IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer.*

[111] Xavier Vilajosana, Kris Pister, and Thomas Watteyne. *Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration*. RFC 8180. May 2017. DOI: `10.17487/RFC8180`. URL: `https://rfc-editor.org/rfc/rfc8180.txt`.

[112] Michele Luvisotto, Zhibo Pang, and Dacfey Dzung. "Ultra High Performance Wireless Control for Critical Applications: Challenges and Directions". In: *IEEE Transactions on Industrial Informatics* 13.3 (2017), pp. 1448–1459. ISSN: 15513203. DOI: `10.1109/TII.2016.2617459`.

[113] Vasuki Narasimha Swamy, Sahaana Suri, Paul Rigge, Matthew Weiner, Gireeja Ranade, Anant Sahai, and Borivoje Nikolic. "Cooperative communication for high-reliability low-latency wireless control". In: *IEEE International Conference on Communications* 2015-September (2015), pp. 4380–4386. ISSN: 15503607. DOI: `10.1109/ICC.2015.7249012`.

[114] B. Kilberg, C. B. Schindler, A. Sundararajan, A. Yang, and K. S. J. Pister. "Experimental Evaluation of Low-Latency Diversity Modes in IEEE 802.15.4 Networks". In: *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. 2018, pp. 211–218.

[115] Craig B. Schindler, Thomas Watteyne, Xavier Vilajosana, and Kristofer S. J. Pister. "Implementation and characterization of a multi-hop 6TiSCH network for experimental feedback control of an inverted pendulum". In: *2017 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOpt 2017* (2017). DOI: `10.23919/WIOPT.2017.7959925`.

[116] Ankur Mehta, Branko Kerkez, Steven D Glaser, and Kristofer SJ Pister. “TDMA-based dual-mode communication for mobile wireless sensor networks”. In: *Sensors* 12.12 (2012), pp. 16194–16210.

[117] Hani Gomez, Daniel Contreras, Joseph Grenspun, and Kristofer Pister. “Zero Insertion Force MEMS Socket for Microrobotics Assembly”. In: *MARSS*. 2017.

[118] Hani C Gomez, Craig B Schindler, Harry L Clark, Joseph T Greenspun, and Kristofer SJ Pister. “Zero Insertion Force MEMS Socket: 3d Multi-Chip Assembly for Microrobotics”. In: *2019 20th International Conference on Solid-State Sensors, Actuators and Microsystems & Eurosensors XXXIII (TRANSDUCERS & EUROSENSORS XXXIII)*. IEEE. 2019, pp. 1732–1735.

[119] Ioana Suciu, Filip Maksimovic, David Burnett, Osama Khan, Brad Wheeler, Arvind Sundararajan, Thomas Watteyne, Xavier Vilajosana, and Kris Pister. “Experimental Clock Calibration on a Crystal-Free Mote-on-a-Chip”. In: *IEEE International Conference on Computer Communications. CNERT: Computer and Networking Experimental Research using Testbeds*. 2019.

[120] Tengfei Chang, Timothy Claeys, Mališa Vučinić, Xavi Vilajosana, Titan Yuan, Brad Wheeler, Fil Maksimovic, David Burnett, Brian Kilberg, Kris Pister, and Thomas Watteyne. “Industrial IoT with Crystal-Free Mote-on-Chip”. In: *to appear in VLSI*. 2020.

[121] Thomas Watteyne, Joy Weiss, Lance Doherty, and Jonathan Simon. “Industrial IEEE802. 15.4 e networks: Performance and trade-offs”. In: *Communications (ICC), 2015 IEEE International Conference on*. IEEE. 2015, pp. 604–609.