UNIVERSITY OF CALIFORNIA
SANTA CRUZ

DESIGN AND IMPLEMENTATION OF A POWER-AWARE DYNAMICALLY SAMPLING WILDLIFE COLLAR

A thesis submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Maxwell J. Dunne

DECEMBER 2014

The thesis of Maxwell J. Dunne is approved:

_________________________________________________________________________
Professor Gabriel Elkaim, Chair

_________________________________________________________________________
Professor Patrick Mantey

_________________________________________________________________________
Professor Christopher Wilmers

_________________________________________________________________________
Tyrus Miller
Vice Provost and Dean of Graduate Studies
# Table of Contents

List of Figures ........................................ v
List of Tables ......................................... xii
Abstract .................................................. xiii
Dedication ................................................ xiv
Acknowledgments ....................................... xv

1 Introduction and Background ....................... 1

2 Implementation ....................................... 6
   2.1 Accelerometer and Magnetometer ................. 8
   2.2 GPS .................................................. 12
   2.3 Data Sampling ..................................... 18
   2.4 Data Encoding ..................................... 21
   2.5 Data Logging ...................................... 24
   2.6 Data Extraction ................................... 28
   2.7 Hardware Evolution ................................. 34

3 Results ............................................... 37
   3.1 Accelerometer and Magnetometer ................. 39
   3.2 GPS Selection and Power Budget ................. 57
   3.3 Data Logging ...................................... 84
   3.4 Data Logging Power Budget ....................... 85
   3.5 Overall Power Budget .............................. 87
   3.6 Biologist Behavior Analysis ..................... 91
4 Conclusion and Future Work

4.1 Conclusions ......................................................... 93
4.2 Future Work ......................................................... 96
  4.2.1 Minimizing Radio Uptime ................................. 96
  4.2.2 Low Power Processor ....................................... 96
  4.2.3 RTCC addition .................................................. 97
  4.2.4 GPS Clock Tuning .............................................. 97
  4.2.5 Onboard Flash memory ...................................... 98
  4.2.6 Behavior Identification ...................................... 98

Bibliography ............................................................. 99
List of Figures

2.1 Sample I2C transaction showing showing a read of an I2C EEPROM from [1]. The transaction starts with the start command followed by the slave address. Note that in this case the bus is running in 15-bit address mode instead of the standard 7-bit with the slave acknowledging between bytes. The master then sends the slave address wanted for reading with the slave acknowledging once again. The slave transmits back the requested data. Once the requested data has been received, the master transmits a NACK followed by a stop command to end the transaction. ............ 10

2.2 State machine of GPS Sampling behavior. Due to the need to conserve power, the GPS receiver is slept most of the time and only fully powered for certain events. If desired at any point it can be commanded to enter active mode with samples being taken continuously until commanded to stop. Outside of continuous mode, the receiver is wakes from sleep on a timeout with samples being taken until a certain number of fixes have been attained or sampling has continued for too long. .................. 14

2.3 Flow chart of the initialization routine of the GPS. With the inherent interface characteristics of the receiver special care must be taken to ensure consistent startup behavior. As the GPS does not parse messages while a command is being processed initialization must wait until the command has been parsed and data flow has resumed. .................. 15
2.4 Flowchart of the recovery process. When an invalid packet ID is found, either a whole sector has been marked as corrupt or data has been lost in some other form and it must be corrected. The process starts by the discarding of the first two bytes. With these two bytes gone, the packet ID is attempted to be read again; as long as there is not a valid packet ID, bytes continue to be removed. With a valid packet ID the second half of recovery begins. It is not enough to use the discovered ID as the start of a valid packet, as IDs are not unique and can be packet data instead. To correct for this problem, the length of the potential packet is used to extract the next potential ID. If this ID is in the valid set of IDs, the recovery process is complete.

2.5 PCB Layout of the multi sensor development kit. The MEMs sensors and microSD card were powered by the UNO32 with the GPS having its own regulator to allow for 3.3V or 5V receivers. Note that the temperature sensor was not present in the original design, and was added manually after board assembly.

2.6 PCB layout of small scale collar tag. This board has only the selected sensors along with accompanying microprocessor. Assembly issues continued within this variation until the introduction of kapton as a stencil material. The board is mainly single sided with the microSD card socket and and inductor for the power supply mounted on the bottom. This power supply proved unusable and an in-line power regulator was built into the cable connector was developed instead.

3.1 Scatter plots of overlayed accelerometer values during orientation trials. This overview already shows the differences between the accelerometers as: in ideal conditions, all three sensors should contain identical point clouds. While the clouds are not identical, given the real world conditions, they are similar enough to indicate that all the sensors are working as expected.

3.2 Scatter plot of the Bosch accelerometer cloud during orientation trials. This zoomed plot shows that the Bosch accelerometer is performing as expected with the denser clouds caused by the sensor being held in that particular orientation for longer periods of time.

3.3 Scatter plot of the Freescale accelerometer cloud during orientation trials. This zoomed plot shows that the FreeScale accelerometer is performing as expected with results very similar to the Bosch.
3.4 Scatter plot of the STMicro accelerometer cloud oriented to show banding issues. With the cloud oriented to align with a single plane, it is clearly showing that the STMicro has a quantization error. While small sections seem to have higher resolution, overall the significant banding makes it unusable.

3.5 Scatter plot of overlayed magnetometer values during orientation trials. This overview already shows issues with both the Honeywell and STMicro being malformed in interesting ways. Both show groupings of points within sub-boxes with borders. Manipulation of zoom levels show the same behavior within each individual box. The reason that both sensors have similar odd behavior, not the reasoning behind the behavior, can be surmised by inspection of the datasheets which show the exact same register layout between the two magnetometers; the assumption is that in some form STMicro copied the silicon from Honeywell.

3.6 The Allan variance and autocorrelation plot generated for the X axis of the accelerometer. Note the nearly flat line of the autocorrelation plot indicating that the noise characteristics generated are correct.

3.7 The Allan variance and autocorrelation plot generated for the X axis of the magnetometer. This attempt at determining noise characteristics fails due to the magnetometer noise being correlated.

3.8 Scatter plot showing the noise correlation between magnetometer values and temperature. Temperature data was taken using an analog temperature sensor overnight to allow for temperature fluctuations in the testing environment. This plot clearly shows magnetometer values increasing with temperature.

3.9 Plots of time to sensor stability in terms of output data rate for the magnetometer and accelerometer. From these plots we can see little gain from sleeping the accelerometer instead of powering it off. The magnetometer shows a slight improvement from sleep versus power off but with only three output data rates taking longer the difference is insignificant.

3.10 Plots showing current draw of the accelerometer in both active and standby modes with the red points being cited datasheet values. Current draw from the accelerometer increased with the output data rate as expected. There is a 20μA offset across all values which is assumed to be an error from the measuring setup.
3.11 Plots showing current draw of the magnetometer in both active and standby modes with the red points being cited datasheet values. Due to oversampling, the actual internal rate of the magnetometer is the output data rate * oversample ratio. With this accounted for, current draw increased with the internal data rate as expected. There is a $20\mu$A offset across all values which is assumed to be an error from the measuring setup.

3.12 Plots showing current draw of the accelerometer over time in both active and standby modes. While current draw in standby mode fits a standard distribution, in active mode, there is skewed distribution with current spikes to significantly larger levels.

3.13 Scatter plot showing cold start position accuracies under ideal conditions. Each receiver was forced to do a hard restart with only the factory ephemeris data enabled, showing the position accuracy after the first 2D fix. In ideal conditions, it is very clear that the uBlox GPS receiver has the greatest accuracy, with the Locosys and MediaTek following behind. Note that for the tracking collar application, it is expected that the GPS receiver will rarely be operating in ideal conditions.

3.14 Scatter plot showing warm start position accuracies under ideal conditions. Each receiver was forced to do a restart with last position, ephemeris, and current time, showing the position accuracy after the first 2D fix. In ideal conditions, during a warm start, the uBlox GPS receiver has the greatest accuracy, with the Locosys and MediaTek receivers following behind but closer than the cold start. During normal collar operation, it is expected that warm starts will be the most common start experienced although, again, not under ideal conditions.

3.15 Scatter plot showing hot start position accuracies under ideal conditions. Each receiver was forced to do a restart with last position, ephemeris, current time, and satellites in view, showing the position accuracy after the first 2D fix. In ideal conditions, it is very clear that the uBlox GPS receiver has the greatest accuracy, with the Locosys and MediaTek following behind. During collar operation, hot starts will rarely be encountered due to the power drain from the GPS.

3.16 Scatter plot of cold start variance versus time to first 2D fix under ideal conditions. In ideal conditions, the uBlox GPS receiver has the greatest number of fast, low variance starts with the MediaTek and Locosys following close behind.
3.17 Scatter plot of warm start variance versus time to first 2D fix under ideal conditions. In ideal conditions, the Ublox and Locosys GPS receivers have low variance positions but several outliers where fixes take substantially longer. In contrast, the MediaTek and San Jose Receivers have a higher variance but no outliers with long fix times.

3.18 Scatter plot of hot start variance versus time to first 2D fix under ideal conditions. In ideal conditions, all receivers tested had tight clustering below 3 seconds except the San Jose which had most of its locks concentrated over the 3 to 4 second interval.

3.19 Histograms of GPS height at first 2D fix after cold start under ideal conditions. All receivers clustered together around true height with the exception of a single spike left of true height; note that the GPS uBlox receiver's spike is substantially smaller than the others. As the collar is power-limited, time to achieve a 3D fix is not available, and thus a 2D fix's height is of limited use.

3.20 Scatter plots of GPS position after cold start, showing how each GPS walked in its location until the trial timed out. It is clear that both the San Jose and Locosys wandered around the first point found, while MediaTek and uBlox corrected to near true and refined estimates from there. While this can be used to infer the number of samples required for an accurate fix, there is little point in performing this test for the other starts.

3.21 Scatter plot showing cold start position accuracies under woods conditions after the first 2D fix. This plot very clearly shows the Ublox GPS Receiver having the greatest accuracy with the rest following the same order as previous trials. Note that the Sirf Star receiver has better accuracy then the MediaTek but its results are more skewed along a single axis. During this test the receivers were operating much closer to collar conditions, and as such, the woods tests had a much greater weight in the selection process than ideal conditions.

3.22 Scatter plot showing warm start position accuracies under woods conditions after the first 2D fix. This plot very clearly shows that the Ublox GPS Receiver continued to have superior performance although it fixed its positions fewer times than the MediaTek or Locosys. The Locosys and MediaTek's high number of fixes is offset by their low accuracy while the San Jose performed comparably to the other receivers.
3.23 Scatter Plot showing hot start position accuracies under woods conditions after the first 2D fix. This plot clearly shows that the San Jose has much worse position accuracy compared to the other receivers with fifty less fixes. The other receivers all achieved the same number of fixes with the Ublox attaining slightly higher accuracy than the others.

3.24 Scatter Plot of cold start variance versus time to lock under woods conditions after the first 2D fix. This plot shows that all the receivers had difficulty in achieving accurate fixes under woods conditions. All receivers tested had significant outliers although the San Jose and Sirf Star had the most.

3.25 Scatter Plot of warm start variance versus time to lock under woods conditions after the first 2D fix. The Sirf Star and San Jose continue to have issues in outliers while the others continue the trend from the cold start with no receiver distinguishing itself from the others.

3.26 Scatter Plot of hot start variance versus time to lock under woods conditions after the first 2D fix. This plot clearly shows that the majority of fixes occurred within 30 seconds across all receivers. The MediaTek and Ublox both have fast fixes with some slower outliers while the Locosys has fewer outliers but a longer minimum time to fix. The Sirf Star outperformed all the others in achieving fixes with no single fix taking more than 10 seconds to achieve.

3.27 Scatter plot showing cold start position accuracies under canyon conditions after the first 2D fix. It is very clear from this plot that the Ublox and Sirf star are not up to the task with fewer than half the fixes of either the Mediatek or the Locosys. The San Jose actually performs better in this case but is still very inaccurate. The Locosys has the most fixes and the tightest circle but has an interesting offset with most fixes to the left of true. Note that the low performers, Sirf Star and Ublox, have the tightest circles of all receivers; it is assumed that the firmware onboard those receivers require greater confidence before they will pronounce a fix.

3.28 Scatter plot showing warm start position accuracies under canyon conditions after the first 2D fix. Unlike the previous cold start trial the warm starts mimic the woods and ideal conditions with the uBlox having the highest accuracy, albeit with 20 fewer fixes than any other receiver. The MediaTek came closest in accuracy to the uBlox but with fewer fixes than either the Locosys or the Sirf Star.
3.29 Scatter plot showing hot start position accuracies under canyon conditions after the first 2D fix. With the additional information available to the receivers, this trial is very similar to trials performed in ideal conditions except for the San Jose which performed far worse in this trial.

3.30 Scatter Plot of cold start variance versus time to lock under canyon conditions after the first 2D fix. This plot shows that all receivers have substantially decreased accuracy of their fixes along with increased times to achieve fix. Of interest is the cluster of fixes from Locosys which fix immediately upon start; this is the same cluster of fixes that have the observed offset in the position plot. It is unknown what causes the Locosys to exhibit this behavior which does not occur in the other trials.

3.31 Scatter plot of variance versus time to lock under canyon conditions after the first 2D fix. With the increased information given by the warm start this plot shows that the GPS receivers are able to achieve fixes with increased time and reduced accuracy. All receivers had significant outliers with MediaTek having the fewest.

3.32 Scatter plot of variance versus time to lock under canyon conditions after the first 2D fix. This plot shows that the increased information given from the hot start are generally able to acquire fixes under ten seconds with decreased accuracy barring the very poor performance of the San Jose.
List of Tables

2.1 Layout of a sample 512-byte sector. The data payload is bracketed by headers and footers while timestamps and checksums ensure additional data integrity. .............................................................. 22

3.1 Overview of accelerometers and magnetometers tested. The first three are triaxial accelerometers with varying sample times and dynamic range. The last three are triaxial magnetometers used for sensing Earth’s magnetic field (used for animal heading information). Note that the STmicro device in a combined accelerometer and magnetometer in a single device which is in some ways easier to integrate than two separate devices. ................................. 39

3.2 Table of noise parameters for the accelerometer and magnetometer. These values are the averages taken from two different overnight trials. Note that the magnetometer noise characteristics are of limited use due to the temperature correlation................................. 47

3.3 Summary of characteristics for GPS receivers tested.................. 57

3.4 Overview of sizes and rates for logged data ......................... 85

3.5 Table of Subsystem usage. Summary of values determined in previous sections along with the total subsystem power usage. ................................. 87

3.6 Table summarizing sensors active time and the total time the processor needs to be active. Even assuming that sensors do not overlap, the microprocessor spends the vast majority of its time waiting. Note that the accelerometer and magnetometer uses an estimate of 1ms per I2C transmission of a 6-byte triplet. ................................. 89
Abstract

Design and Implementation of a Power-Aware Dynamically Sampling Wildlife Collar

by

Maxwell J. Dunne

Animal tracking collars have proven incredibly useful to biologists providing insights into range, physiology, and environmental interaction. Current collars rely on scheduled sampling of GPS that must be preset at deployment (though advanced ones can modify the schedule through radio communications). This work details the design and development of a third generation of Mountain Lion tracking collar that can identify animal behavior in situ using advanced onboard sensors. The power-aware collar relies on triaxial accelerometers and magnetometers and signal processing techniques to classify behavior and dynamically scale its sampling for higher fidelity during interesting periods and sleeping power hungry sensors during uninteresting ones. An analysis of sensor performance and down selection is presented, along with software and hardware required to implement the collar. The resulting collar is capable of sampling its primary sensors at a high data rate (50Hz/5Hz respectively) and still last over three years on a typical battery using a low power processor. Data is logged to an onboard microSD card, and several steps have been included to ensure data integrity and recovery in the case of faults in the microSD card. It is shown that the largest power draw on the system will be the VHF/UHF radio link, and that the embedded processor on the collar should be used to minimize the time spent transmitting data by using a dedicated symbol table to encode behavior and positions for interesting events in the animal life.
To my Queensland Blue Heeler,

Ruby WiseChoice Bandit,

accompanying me through thick and thin until nearly the end.
Acknowledgments

I started this project at the same time I entered graduate school, unaware of the true scope of the work. Without the assistance of those around me I would have never finished.

My advisor, Professor Gabriel Elkaim, for taking a chance and letting me enroll into Mechatronics without the proper prerequisites as an undergraduate. And since then pushing me to achieve new heights and never slow down. My readers, Professor Patrick Mantey and Professor Christopher Wilmers, who provided guidance along the way.

My family, supporting me in their myriad ways. My mother, always willing to lend an ear for any issue. And in my time of need, her vast knowledge of grammar and willingness to edit what she did not understand proved invaluable in finishing my thesis. My father for always providing food and support in times of need. My sister and her kids, for nothing cheers you up quite like playing with a two-year old. To all the others, thank you for being there when I needed you.

The ASL and its inhabitants, for always providing a home away from home. While there were tumultuous times, the good always outweighed the bad. Headphones were key for productivity, but removing them always provided a refreshing break.

The Mechatronics class, while the class took far too much of my time it provided invaluable experience. So much of my debugging expertise comes from helping students trying to fix their robots at 3AM.

To all the others I might have missed, thank you for your support.

This work was generously supported by the National Science Foundation, DBI-0963022. We wish to thank them for the opportunity to work on a truly interesting, real world problem.
Chapter 1

Introduction and Background

Large mammals play a significant role in shaping their prey populations and environment. Despite the effect these animals can have, there is little data regarding their physiology and daily behavior. This is primarily due to the difficulties of observing these mammals in the wild. As such, there is a great need for *in situ* data collection that does not disrupt the animals’ behavior. As a method of collecting data on the animals’ behavior, GPS-based tracking collars have been used to record time and location to study populations of animals. The intention of the tracking collar presented in this work is to provide long-term data regarding metabolic requirements, physiology, movement, and foraging patterns of Mountain Lions.

The study of Mountain Lions has a long and distinguished history, with wildlife researchers using more and more sophisticated technology to tease out details of the animals’ lives. Early efforts included remote cameras triggered by infrared beams or pressure plates, gathering fleeting glimpses into the animals’ behavior (i.e., nest predation, [2], [3]; feeding or resting area utilization, [4]). Later attempts were made to fit sensors directly to the animals (e.g.: collars). Unfortunately, the size and weight of the sensor packages, combined with the limited battery life and communications capabilities, hindered the measurement of energetics and other physiological details of terrestrial mammals. Even today, the vast bulk of deployed
collars simply measure GPS position at fixed intervals.

The enabling technology to improve animal energetics comes not from biologists, but rather from demands in the telecommunications industry. Technology originally developed for airbag deployment (MEMS accelerometers) and laptop hard drive protection (magnetometers) have found applications within the Smartphone market. Within the past decade, industry has created low-power and low-cost accelerometers and magnetometers which provide an alternative means of sensing animal behavior and metabolic conditions. By calibrating the sensor readings before deployment, these sensors have been used with great success to gain insight into the energetics of marine mammals [5]. By utilizing many of these same techniques and technologies, it is hoped that a similar level of success can be obtained with terrestrial mammals as well.

Biologging devices already play a large role in the study and conservation of wildlife. GPS equipped collars have allowed scientists to collect thousands of data points throughout the lifespan of a collar and have provided insight into range, habitat, and migration patterns [6]. While these devices provide great detail into movement patterns, they provide little information on the animal’s physiology and behavior. Previous generations of the collar were developed that took full advantage of the low power accelerometers in order to gain great insights in these areas.

The first generation of the collar [7], [8] combined a GPS and accelerometer with a microcontroller. GPS samples were taken at a user defined interval with accelerometers being sampled at 64Hz. The electronics were ruggedized into a collar via Telemetry Solutions. Tag calibration was completed using domesticated dogs. Four Mountain lions were collared with the prototype devices in the Santa Cruz Mountains. Due to manufacturing flaws, three of the four collars failed to recover any data. The success of the fourth collar prompted the creation of the 2nd generation tracking collar.

In order to address durability concerns, the second generation of collars [9],
were built with the help of Vectronic Aerospace, a company that had previously succeeded in building commercial tracking collars. To ease integration of this collar, custom electronics were designed to fit within the battery housing of a commercially produced tracking collar. The electronic hardware had been modified with the addition of a magnetometer. This piggybacking of the battery casing caused issues since the custom electronics could not communicate with the collar systems provided by Vectronic. Therefore, there was no ability to alter the GPS sampling regime or to report back behavioral data via the radio link. This issue was especially problematic in tracking kills since without behavioral data confirming a kill had taken place, significant effort can be wasted in surveying potential kill sites. This thesis is concerned with the development of the third generation of collars designed to partially mitigate these issues.

The third generation addresses the issues of the previous generations by designing a fully integrated collar. While Vectronic is still responsible for housing the electronics and providing an interface for the radio link, the rest of the collar is a custom designed fully integrated collar; the details of which are set out in this work. The collar uses onboard storage to record GPS, accelerometer, and magnetometer data for off-line processing of high fidelity data, while determining behavioral data to augment the conventional radio telemetry from the collar. With on-board processing allowing for an adaptive sampling routine, power can be conserved by shutting down systems as needed; greatly extending collar deployment times.

This thesis is broken down into two main parts: Chapter 2: the design and implementation of collar hardware and software; Chapter 3: results from testing sensors for usage along with an energy budget analysis.

Section 2.1 concerns the development of a robust system for accessing the MEMs sensors. With an I2C interface and a generic hardware drivers, individual drivers for each sensor were developed allowing for easy access to the full range of the sensors’ capabilities.
Section 2.2 covers the development of GPS software driver. GPS receivers are serial devices which are unable to return position fixes within a deterministic time interval. These high-power devices will drain battery life if left on and require a power-aware state machine to cycle between sleep states and active position acquisition.

Section 2.3 concerns the development of a centralized sampling system. With a variety of sensors needing to be sampled at variable rates, a software architecture using hardware timers was developed that can gracefully switch sampling rates of any individual sensor on-the-fly.

Section 2.4 concerns the development of a data collator. Various data sources must be aggregated for efficient packing on the storage medium with allowances for data integrity (CRC).

Section 2.5 concerns the hardware interface to the microSD card. Keeping a filesystem readable by standard PCs allow for more robust data recovery in the event of data corruption. Conversion of standard SPI-SD blocking code to a polled topology allows data to be written without impacting other collar operations.

Section 2.6 concerns post extraction of the data. Python scripts allow integrity checking of data with recovery procedures to minimize data loss in the event of SD card corruption.

Section 2.7 covers the various hardware developed over the course of the Power Aware Collar development.

Section 3.1 concerns the selection and noise characterization of accelerometers and magnetometers. Testing eliminated non-viable sensors and resulted in a down-selection of the magnetometer and accelerometer used for the collar. Noise characteristics were determined along with power drain and startup performance. Sensor lifespan is calculated from a simple duty cycle argument.

Section 3.2 concerns the selection of the GPS receiver and power consumption. Several models were tested over a variety of start types and environmental conditions. With a GPS receiver selected, energy budgets were calculated using the
GPS sampling state machine from Section 2.2.

Section 3.4 estimates the power budget of the microSD card. Using the sampling regime previously developed in Section 3.1 and 3.2, energy budgets were calculated using static sampling as a conservative estimate.

Section 3.5 estimates the total power budget and lifespan of the collar. This is done using both the current high-power processor and the low-power processor intended for the next iteration of the custom electronics.

Section 3.6 summarizes the work done by Dr. Y. Wang in using the second generation of collars to identify Mountain Lion behavior using random forests.

Chapter 4 presents the conclusions and future work necessary to bring the power aware collar to a state ready for full deployment on the mountain lion population.
Chapter 2

Implementation

This chapter concerns itself with the hardware and software of the Mountain Lion Tracking Collar. With any project there are significant hurdles in ensuring that all parts work smoothly. With the collar there are two main goals of the software: the interfacing and sampling of a variety of sensors; and the manipulation and long term storage of sensor data.

This project is an attempt to gain additional insight into behavior and, as such, certain requirements of the sensor interfaces must be met. The need for dynamic sampling requires that the complete suite of the Accelerometers and Magnetometers abilities be readily available. Due to the power constraints inherent in any battery-powered device, special care must be taken with the GPS receiver. The GPS, with its high power draw and non-deterministic sampling time, must be interfaced in such a way as to both conserve power and to generate position information samples as required. A sampling architecture is required to ensure accurate sampling across all sensors.

With the ability to sample sensors, care must be taken to ensure that the data remains intact. On-board storage of data is limited and efficient packing of myriad sensor outputs to ensure a sufficient lifespan of the collar. The scarcity of this data makes it essential for data integrity checks to be implemented on the collar. To ensure recovery in non-ideal conditions, storage mediums should be robust and be
able to operate independently of the collar. At the same time, these data logging operations must not interfere with other systems on the collar. Once the data is retrieved, care must be taken to minimize data corruption.

Given all these software requirements, customized hardware must be designed for testing and development.
2.1 Accelerometer and Magnetometer

In order for the biologist clients to infer greater insights into animal physiology, they must be able to process high fidelity data from the onboard accelerometer and magnetometer triads in order to determine animal behavior; this is a central requirement of the mountain lion tracking collar system. In order to meet these objectives, sensors must be sampled at different rates depending on the environmental conditions and current animal behavior. The sensors themselves come from the Smartphone industry and, as such, have their own interface characteristics.

MEMs Sensors of this nature have few interface methods due to their intended usage; in general this is either I2C or SPI as the bi-directional communication method. Due to size constraints, the pins for these communications channels normally overlap, allowing for only one to be operational on a single board design. These sensors typically have additional interrupt pins to allow notification of the microprocessor for asynchronous events.

I2C was originally developed for Phillips semiconductors in the 1980’s as an internal bus. It was created as a way to interface several low-speed peripherals and is now commonly used to interface microprocessors with multiple devices using minimal hardware. SPI was developed by Motorola (now Freescale) alongside the first microprocessors in 1979.

I2C and SPI are both serial communication links with their individual advantages and disadvantages. SPI is capable of full duplex communication and can be clocked in the Mhz range. At the same time, it requires additional control lines and a separate slave select pin for every sensor. Due to its dual shift register nature, it requires little on top of the hardware specification in order to establish communications. I2C on the other hand is a more complex communication architecture due to its half duplex nature which requires a handshaking protocol. It is also typically clocked at a much slower rate (nominally 100Khz, with fast mode being only 400Khz). The major advantage of I2C is that it is only a two-wire hardware bus which allows a large number of devices to be attached without ad-
ditional hardware. This advantage, paired with the need for testing many devices, along with the option of expanding the sensor suite, made I2C the obvious choice for the collar.

The I2C has requirements in both hardware and software that must be addressed for a power-aware device. The hardware for I2C consists of clock and data lines with independent pull-up resistors. When a device wishes to communicate on the bus, they ground the line causing a zero on the bus. The master controls the clock line which idles high. When this clock is pulled down the device has until the clock strobes back up to ensure that the data is transmitted. This open-drain configuration necessitates the need for calculating the ideal pull-up resistor size. Too small of a resistor and too much power will be wasted during communications, while too large of one will round the edges of the signal and force a slower clock speed, resulting in power being wasted while communicating.

The protocol that runs on the I2C bus is a master-slave configuration. This means that without initial communication from the master the slave devices will remain silent. When the master wishes to communicate with a device as shown in Fig. 2.1, it starts the transmission by sending a *start* condition by grounding data while the clock remains high. With this transmission complete, the master chooses the slave to communicate with by sending out the slave’s unique 7-bit address, along a write or read command as an additional bit to form a complete byte. The master then waits for acknowledgement from the slave (slave pulls date line low). Once the ack is received, the master then transmits the desired command and/or address and waits for another acknowledgement. The slave’s acknowledgement starts the transmission of the data in byte chunks. At the end of every byte, the master can send a stop command or a repeated send command. If the master transmits a repeated send, the device continues the current command with the next register in sequence. This is used to great effect within the driver, as the data most frequently wanted from the sensor is in the form of a 6-byte triplet for XYZ data. Using the repeated send allows the microcontroller to retrieve the
Figure 2.1: Sample I2C transaction showing a read of an I2C EEPROM from [1]. The transaction starts with the start command followed by the slave address. Note that in this case the bus is running in 15-bit address mode instead of the standard 7-bit with the slave acknowledging between bytes. The master then sends the slave address wanted for reading with the slave acknowledging once again. The slave transmits back the requested data. Once the requested data has been received, the master transmits a NACK followed by a stop command to end the transaction.

Due to the complication of this transaction, the I2C driver is blocking. While it would be possible to write an interrupt-driven state machine, the benefits of doing so would be small since the goal of the collar is to take this sensor data. With the time sensitive modules driven by interrupts (i.e. serial) there is little loss from blocking during the very short I2C transactions.

Once a generic low-level I2C driver was written, individual sensor modules could be developed. These were written with high level functions in mind to allow easy access to the sensors’ settings and data (i.e. setting sample rates). With a customized module for each sensor relying on a single hardware driver, issues from differing hardware configurations were generally mitigated.

Verification and validation occurred during operational usage. The driver was originally written during the testing of sensors and to date has been used continuously. Slight modifications have occurred, such as the addition of triplet reading, but the core code has handled all requirements. With this long operational usage,
it is concluded that the I2C driver, along with its accompanying sensor drivers, will continue to perform well.

Accelerometer and magnetometer data remains essential to determining sample rates and recording interesting behavior of the mountain lions. These sensors, connected over I2C, have several communication hurdles. The implementation of a generic blocking I2C driver allowed several sensors to be integrated easily with little impact to collar operations. Individual drivers for each sensor allow the full capabilities of the sensor to be used, including on the fly reconfiguration. Verification and validation over long term usage ensures that the I2C interface will enable access to the sensors over the lifespan of the collar, allow for hardware improvements and sensor upgrades into the future.
2.2 GPS

The primary requirement for the mountain lion tracking collar is to *track* the animal, hence the requirement for GPS positioning. The position data from the collar is used by the researchers to determine not only range but estimates of energy use as well. To yield the appropriate information, GPS should sample at set time intervals along with additional opportunistic samples when interesting events occur. Unfortunately, this process is problematic due to the specifics of GPS receivers.

As previously stated, GPS receivers are used in such a way that makes adhering to the required sampling regime challenging. Specifically, a GPS receiver will not communicate a valid position fix in a deterministic time interval; e.g.: a GPS receiver must first acquire a valid satellite constellation before it can determine its own position. As such, while reasonable bounds can be placed on the time it takes to generate a valid position, it is highly dependent on environmental conditions and the specifics of the underlying GPS chipset. Additionally the interface to the GPS device is a serial link, and it is thus not available in a polled mode; that is, the processor must wait for the GPS receiver to communicate its messages and not vice-versa (this can be thought of as the hollywood model: don’t call us, we’ll call you). The GPS receiver is configured such that messages are transferred at set intervals, but these are slaved to GPS time not processor time. This means that the processor must wait (often several seconds to possibly minutes) to validate the GPS message. These complicating factors make a simple GPS sampling software difficult to implement in any sort of power-aware fashion; the GPS receiver is the most power-hungry sensor on the collar and as such its ON time must be minimized to ensure long collar life. To mitigate this complexity a state machine is the ideal choice for managing the GPS receiver. Essentially the GPS sampler is always within one of two distinct states: fixed interval sampling and opportunistic continuous sampling. The state machine allows the GPS software module to cleanly switch between these two states while avoiding complex
switch logic to implement the same. Additionally, the state machine gives the software flexibility and extensibility should future changes become required. As the GPS communicates new information on set time intervals, this also allows the GPS state machine to run inside the existing sampling framework significantly simplifying implementation.

A graphical representation of the state machine is shown in Fig. 2.2. The state machine starts by initializing the GPS receiver, a flow chart of which is displayed in Fig. 2.3. Initialization is both a delicate and lengthy procedure, however is only performed at collar startup and the information retained within the GPS receiver firmware for the rest of the collar deployment. There is an ambiguity about the original state of the GPS receiver; to mitigate this ambiguity the initialization procedure is written in such a way as to always work regardless of the original state of the GPS receiver. To minimize power consumption we raise the baud rate of the communication channel to the highest available, thus minimizing the time either the receiver or the microcontroller must spend actively communicating. Initialization commences with a microcontroller initiated command to set the baud rate to 115200 which is ignored if the receiver is already at that speed. The initialization process resets the UART peripheral to the higher baud rate flushing the serial buffer to ensure there are no message remnants remaining. At this point the microcontroller monitors the UART for a valid start of message. Once received, the GPS receiver is now confirmed to be at its maximum baud rate and the other configuration messages can be sent. The UART is flushed and waits for a valid start message between each configuration message ensuring that the GPS has processed each command. At the end of this process, the GPS receiver is fully configured and is put into sleep mode (low power) pending further sampling requirements. In order to avoid reconfiguring the GPS receiver multiple times, the GPS receiver is slept rather than turned off as the power consumed by the GPS receiver in sleep mode is minimal. Note that the sleep function is enabled by toggling a pin on the microcontroller.
Figure 2.2: State machine of GPS Sampling behavior. Due to the need to conserve power, the GPS receiver is slept most of the time and only fully powered for certain events. If desired at any point it can be commanded to enter active mode with samples being taken continuously until commanded to stop. Outside of continuous mode, the receiver is wakes from sleep on a timeout with samples being taken until a certain number of fixes have been attained or sampling has continued for too long.
Figure 2.3: Flow chart of the initialization routine of the GPS. With the inherent interface characteristics of the receiver special care must be taken to ensure consistent startup behavior. As the GPS does not parse messages while a command is being processed initialization must wait until the command has been parsed and data flow has resumed.
Now that the initialization is complete the GPS sampling state machine enters the interval sampling mode. The first step is to generate its first valid location fix. This is done by first waking up the GPS receiver from sleep and setting the sample rate to 1Hz. Note that the GPS messages arrive piecemeal and asynchronously, and need to be assembled and parsed for processing. This occurs through a combination of interrupts and polled functions; serial messages from the GPS are received via an interrupt which loads them into a circular buffer. In the main loop the GPS handler is called which parses all new characters in the circular buffer. Once a complete message arrives it is parsed and the new data is loaded into the GPS struct. With every new sample, the GPS struct is checked for a self-reported valid fix. A fix indicates that the GPS receiver has locked on to at least 3 satellites and has a valid 2d position. While a 3d position is more accurate, in non-ideal conditions there are no guarantees that the receiver will be able to lock onto at least 4 satellites. If a valid fix has occurred, the position is logged and the successful fix counter is incremented. If there is no valid fix, then the bailout counter is incremented instead. When the user-determined number of valid fixes is reached, the GPS system is put back to sleep. If the bailout counter reaches a predetermined threshold, the GPS receiver is determined to be incapable of tracking its position at the current time and it is put to sleep resetting the interval sampler. That is the GPS state machine now waits in sleep mode for the long interval before starting over again. At any time, the continuous sampling regime can be enabled. Upon entry into continuous sampling the GPS powers up and the sample rate is changed to 1Hz. At this point, regardless of fix, the state machine will log a GPS sample once per second until a stop command is received. Upon receiving the stop command the GPS sampling state machine reverts to its interval sampling mode and the process starts again. Restoring the original timeout value allows interval sampling to continue unimpeded after continuous mode.

Verification and Validation of the GPS state machine was conducted to ensure proper functionality. A test harness was used with fake events to test all state
transitions. Timeout tests were performed in the lab (no valid GPS fixes) and verified the state machine would handle bailouts correctly. Finally the interval timing of samples was reduced to allow sampling outdoors with valid GPS data. With each of these tests the data files were inspected and it was determined that the GPS state machine was performing as required. Position data of the collar remains the paramount information required to infer animal life cycle characteristics and ecology. GPS, due to its non-deterministic time to fix presents challenges in a power-aware wildlife monitoring setting. In order to mitigate these challenges a state machine implementation of the GPS sampling regime was developed and tested, demonstrating that both interval sampling and opportunistic high rate sampling can be achieved. Verification and validation techniques were used to ensure that GPS can be adequately sampled while preserving a maximum battery life on the collar.
2.3 Data Sampling

One of the primary goals of the mountain lion tracking collar is to adaptively sample sensors. Differing sensors require different sample rates which must be adhered to in order for behavior to be recognized. At the same time, a particular sensor’s sample rate must be able to be changed on the fly to account for different behavior and environmental conditions.

To implement the behavior required, time must be kept independent of active processor duties and, as such, hardware timers from the microprocessor are used. The time scales required are both a fast timer (i.e. accelerometer) and a slow timer (i.e. GPS). With the accelerometer being sampled at the highest rate, the fast timer is always slaved to the accelerometer, with other sensors sampling at multiples of that rate. The slow timer is set to a second interval to allow for slow sampling sensors such as GPS to be sampled at hourly scales. Initial implementation of the sampler module used two hardware timer modules running independently. Unfortunately this had an issue in that over time the second timer would lose ticks in comparison to the fast timer. The module was rewritten to only use one hardware timer. This introduced the complication that the second timer was now a multiple of the accelerometer timer. Instead of being able to increment the second timer in the timer interrupt, it must instead increment after a certain number of accelerometer ticks. This caused the overflow counter for the second timer to be altered every time the sample rate changed.

In the new architecture, several changes must occur when a sensor’s sample rate is changed. Upon receiving the new sample rate, the Data Sampling module calculates the new number of ticks required between samples and stores this and the current tick count in memory. Each iteration of calling the Sampler uses the stored tick count to determine if the required interval has elapsed; if so, it performs the required sampling and stores the new current tick count. If the accelerometer’s rate is changed, all sensors attached to that timer have their time intervals changed as required to maintain the same sampling rate. Sensors attached to the second
timer do not require any changes since the second timer will already have been modified.

With the overall architecture of the sampler in place, a selection of the magnetometer and accelerometer sample rates were chosen from the sensors’ capabilities. Upon inspection it was discovered that for every usable accelerometer rate there was a magnetometer sample rate 10x slower. As there is not a need for a high sample rate for the magnetometer, the collar can be set to always sample these two sensors at the 10:1 ratio. This allows the sampling of these sensors to be simplified.

With a sensor sampled, the data must be passed on to the Data Encoding module. To allow for identification of the data, a packet structure is used to wrap the data. Packet formation is simple since packets are statically fixed at compile time, and data integrity is handled within the Data Encoding module. Each packet is given a unique 2-byte ID which precedes each packet submitted. As the data extractor has knowledge of the specifics of the defined packets there is no need for additional identifiers. At the same time as the data is submitted to Data Encoding, relevant data can be submitted to other modules for in situ on-board behavior analysis as needed.

Validation and verification techniques were used to ensure the integrity of the module. Test harnesses were written to ensure that the hardware timers were properly scaling given various sample rates. Each sensor in use was tested at various sample rates and confirmed packets were being submitted to Data Encoding. With all sensors sampling independently, tests were performed in the lab to ensure that all sensors sampled correctly together over long time scales. Sample times were then shortened and tested outside of the lab to ensure integration outside of the lab. With all these tests, data files from the Data Logging module were analyzed to confirm correct operation.

Ensuring correct sampling of sensors is of paramount importance to the collar. This is hampered by the need for accurate sampling with on the fly rate changes.
Software modules using hardware timers were developed to implement the varying sensor rates required. Inadequacies with the hardware timers forced a single timer to be used for all time scales which caused certain software complexity to emerge. Sensors can be commanded to change sample rates at any time with all changes required to the hardware being handled by the module. With this architecture in place, a 10:1 sample ratio was chosen for the accelerometer and magnetometer due to device output data rates having the same 10:1 ratio and in order to allow simpler sampling. A 2-byte ID was prefixed to samples before submittal to Data Encoding to allow for easy extraction. Validation and verification techniques were used to ensure that Data Sampling would provide accurate samples over the lifespan of the collar.
2.4 Data Encoding

One of the primary goals of the mountain lion tracking collar is to take data from a variety of sources and store it for later analysis. Data integrity is essential as corrupted data is not only lost, but determining precisely which data has been lost is a difficult and complicated process. Further, a variety of sources generate data at various rates and these must be combined for efficient packing onto the storage medium in such way as to be easily retrieved later.

Packets from other software modules are submitted to the data encoding module in arbitrary sizes. To ensure that data encoding does not need additional information from higher levels, the interface functions to the module used are instead wrappers to functions using #defines. This allows the module to use `sizeof` to grab the size of the incoming packet instead of using an additional argument from the other module. One consequence of this method is that all packet sizes must be known at compilation time, removing the ability for dynamic packet sizes. However, due to the fixed nature of the sensors generating incoming data, this is not anticipated to cause difficulties.

When a module needs to submit data for encoding, it calls `DataEncoding_SubmitData` with the structure as the only argument, which starts the data encoding process. The first step is to copy the data from the struct to allow the calling module to release that memory, if desired. If the incoming packet fits within the current sector, the process is over at this point. If the packet size is equal or larger than the remaining space in the sector, it copies what will fit into the sector. The module then calls the data logging module on this newly full sector to start the process of transferring the sector to the storage medium. It finishes the process by moving to the next sector and copying the remaining bytes of the packet to the new sector. Once data logging is done transferring the data, the sector is once again available to queue more data. As such, as long as the queue in the sampler does not overflow, there will be no loss of data from this stage.
Table 2.1: Layout of a sample 512-byte sector. The data payload is bracketed by headers and footers while timestamps and checksums ensure additional data integrity.

Data encoding size is dependent on the hardware used for logging, which in this generation of collar is 512 byte sectors. This size is reduced further by the addition of several features for data integrity as shown in Table 2.1. These data integrity sections total 8 bytes in size. Given a sector size of 512 bytes, there is a 1.5625% loss per sector. This is an acceptable loss in exchange for the needed data integrity.

The first level of data integrity is the header and footer. These consist of preset 2 byte numbers at the beginning and end of each sector, which allows the extractor to properly align on sectors. With $2^{16}$ possible values the chances of random data being a sector length apart is quite small.

The second level is the timestamp, which immediately follows the header. The timestamp is 2 bytes filled with the time in seconds from the sampler module. Whenever a new sector is started, this value is immediately written to the sector. This time is checked during extraction to ensure that it is always increasing; it is not necessarily contiguous as sampling conditions can result in either repeats or gaps. If it is not increasing, however, it is an indication that either sectors are out of order or there is a much worse error in the sampler module.

The final layer is a cyclic redundancy check (CRC) which allows later verification that all bytes within the individual sector are correct. The CRC is calculated using DMA hardware on board the microprocessor. This allows for fast compu-
tation time, without loading the processor. The CRC itself is 2 bytes long and uses the CRC-16 polynomial of 0x8005 from [11] following recommendations to use standard CRC polynomials. At the same time, the CRC used does not follow the full guidelines for CRC-16. Although it has the standard seed of 0xFFFF, there are steps after the CRC generation to match the CRC-16 standard which are not supported by the DMA hardware and would thus require additional processor cycles. Instead, the raw CRC is recorded into the sector before the footer and the data extractor handles the necessary modifications to the algorithm.

Verification and validation techniques were used to ensure that all parts of data encoding were performing as required. Test harnesses were written to simulate incoming packets of a wide array of arbitrary sizes with debug messages ensuring that all data was copied into the sectors without losing bytes. The DMA algorithm was tested with a large input set of sectors and compared with known CRCs from data extraction. While this does not prove that the CRC is correct according to standards, it is correct from end-to-end within the collar encoding systems and data extraction. With these tests complete, data encoding was integrated into the collar and large amounts (1.8GB) of both fictitious and real data were taken. These were then analyzed and it was determined that there had been no lost packets. The probability of error in the CRC checking is 0.00001525878%.

Data encoding remains a requirement of the mountain lion tracking collar. With data varying in source and size, the need for a system to stitch packets together for effective packing was paramount. Through the use of #defines for packet size and a sector queue to hold data, the data encoding module fulfilled these requirements. With several layers of data integrity, including headers and CRC, the ability to determine invalid data and make the necessary corrections was assured. Performing extensive validation and verification techniques on both real and fake data ensures that the data encoding module will provide efficient packing and integrity while minimally loading the available hardware.
2.5 Data Logging

The mountain lion tracking collar would be of little use if it lacked the ability to store the data collected. Although there is a radio link, it’s high-energy usage and low bandwidth makes it problematic for high fidelity data transfer. With the radio link unavailable, most of the data will need to be parsed after collar retrieval which in turn creates the need for large onboard storage. In addition to large capacity, the storage system must be durable. The collars face outdoor conditions, along with the lions’ own great desire to remove the collar. Given the difficulty of acquiring the data in the first place, loss of data due to a damaged collar is simply not acceptable. A simple solution is a robust data storage medium that can be read independently of the collar.

The storage medium which fulfills these requirements is a standard microSD card. These have a small form factor; with the advent of the smartphone they are cheap, and come in a wide variety of storage sizes. Mounted in a standard socket and tacked in place with glue, they are more than durable enough for the lifespan of the collar. In the event of collar breakage, data integrity is preserved unless the card packaging itself is destroyed. Upon retrieval, the microSD card can be read on any PC.

While a PC has built-in drivers for reading the microSD card, the microcontroller must interface the card differently. The standard mode for interface is SD mode which is only available by joining the SD association; this costs thousands of dollars annually. SD mode, a multi-channel high speed system, is not required for the purpose of the collar; SD-SPI mode is used instead. This is a subset of SD mode with only one channel and a maximum clock rate of 20Mhz. The rate at which data is gathered is well below this limit and, as such, SD-SPI is perfectly suited for the collar’s needs.

The microSD card by default does not support SD-SPI upon startup. It must first be initialized into SD-SPI mode, which is done by using MicroChip’s Storage system library [12]. Once this process is complete, the same library can be used
to initialize the FAT filesystem and to use the card in a polled mode. With the filesystem ready, several operations are performed:

1. Current contents of the card are enumerated, looking for an unused filename.
2. A new file is created and its file handler retrieved.
3. The first sector’s address of the file is extracted and stored for use in logging.
4. The file is set to a large size and sectors are allocated to it.
5. The file is closed, forcing a flush of the filesystem to the card.

When this process is complete, the card can be removed, or have power cut at any point, and the only data loss would be the sector currently in progress. Additionally, with an appropriately sized file, there is no need to touch the filesystem again. This ensures that while there might be garbage data at the end of the file, there will be no data lost due to an improperly closed file.

Once the filesystem is setup, logging data is straightforward. Upper level modules submit a pointer to the data logging module pointing to 512 bytes of data needing to be written. To conserve processor cycles in copying memory, the data logging module never copies the data; therefore, the upper level module must not modify the memory until the transaction is complete. At this point, the module will start the SD-SPI transaction. No further action is required by other modules to complete the write. During logging operations, the module checks the error status of each sector written. If one of these sectors fails to write, the logging process halts and attempts to re-initialize the hardware interface of the card. During this time the collar can’t process any more data in the current generation due to the SD-SPI initialization routines being blocking; without the ability to store data, there is little point in taking more data. Future implementations of the collar will include dual data buffers and non-blocking code such that data sampling will continue while the SD card is reinitializing.
The major issue in data logging is in ensuring that data logging operations do not interfere with the sampling of data. While there is significant handshaking at the beginning, the majority of the transaction is clocking data out of the SPI bus. As a result, interrupts are of limited usage and DMA was implemented.

To implement DMA and remove the blocking sections of code, the existing sector write code from Microchip had to be completely rewritten. This was done by turning the blocking function into a hierarchial state machine. The super states come from the original sector write function and, it is assumed from the framework given, at some point Microchip had also been planning to write a non-blocking form of the functions. These super states were each separated out into their own functions and converted to state machines. The lower level functions which did not warrant a full state machine were rewritten to return free or busy with every call rather than blocking code. Once the polled handshake is completed, the DMA transfer starts and writes out the full sector. After a short cleanup transaction the module is ready to write the next sector. At any time upper level modules can query data logging to check whether or not logging is available. With this architecture in place, the polling function can be called continuously at the bottom of main to handle logging functionality since tasks are performed only as needed.

Validation and verification of this subsystem were performed to ensure functionality and that the ability to perform at high speed. A test harness was built that logged sectors to the card at the maximum rate possible over long intervals. The files generated in this manner were then checked via the PC to ensure that the data contained matched what was written. Additionally, tests were conducted to confirm that data logging also performed as required with data coming in at both known and unknown intervals at operational rates.

Data logging on the collar is required to allow high-fidelity processing of sensor information. MicroSD cards, due to their generous storage space and durability, are an ideal choice for the collar storage medium. Combined with the FAT-16 filesystem, the card allows for recovery of data in almost all conditions. To meet
the requirement of minimized logging time, a re-write of the sector write to non-blocking form enabled the collar to perform other duties even while logging at maximum rates. Verification and validation techniques were then used to ensure that data logging remained robust throughout differing conditions for the lifespan of the collar.
2.6 Data Extraction

Without the ability to extract the high fidelity data from the collar, there would be little gained from the deployment. In order to save space and power on the collar, the data is stored in a binary format (which must be extracted to a CSV form for the end user). The data packing to reduce sector waste requires a smart extractor since the data is not necessary continuous. This also creates the need to ensure data integrity and that data lost to corruption remains minimal. As such, a tool that meets these requirements and provides a usable interface is required.

Python was chosen for the data extractor programming language for a variety of reasons. First and foremost is the cross-platform compatibility, which allows the biologists to extract and use the data on any platform where Python is installed (Mac, PC, Linux, etc). While a compiled language could have been used, the ability to compile Python renders the compiled languages’ speed advantages moot in this situation. C, the language the collar runs on, was disqualified on the basis that the differences in platform and operation would make code porting unwieldy with little to no gain. Python is a very feature rich language which conveniently provides all the tools required to implement extraction.

Extraction starts with the parsing of a configuration file of packet IDs. This file contains packet entries in the following format: ID, which also serves as the index for the packet; String Name, a human readable string containing a descriptive name of the packet; a struct used by Python along with its length; and a string containing names of each of the fields. This file is read into a dictionary of dictionaries and is used to extract the data while parsing. Along with functions to read in this data, there are functions for creating new packets to ensure compatibility in the configuration file.

Once the packet dictionary is built, generation of the output file begins. This starts with allocating space for the file ID. This file ID consists of the first valid GPS location found in the data file, which can not be determined without parsing
the file. Instead of having to parse the file partially and restart, the file pointer is moved to ensure blank space large enough for the ID. After this, the packet comment strings are written to the file, one per line. This ensures that each line can be correctly identified without resorting to the source files or packet configurations.

The source file is read in 512 byte blocks to conform to the sector boundaries required by the microSD card. The first operation performed on each sector is verifying the header and footer. If header and footer are not in their proper places, the sector is skipped and the parser advances. With the sector confirmed, the CRC of the sector is calculated, confirming the individual bytes. A correct CRC allows the extractor to append the sector contents to the packet queue.

The data is parsed from this queue until the queue length is less than the packet ID length (two bytes). The first step in parsing is to extract the first two bytes as an unsigned integer which represent the packet ID. This packet ID is then used to look up the parameters for the particular packet in the dictionary. Knowing the parameters, the length of the queue is checked and, if it is too short for the packet length, the process is halted until more bytes are available. With enough packets available, the struct library is used to extract all fields of the packet and write them to the output file in a tab delimited fashion. This process repeats itself as long as the first two bytes read are a valid packet ID.

When an invalid packet ID is found, either a whole sector has been marked as corrupt or data has been lost in some other form and it must be corrected. Due to the missing data, and the nature in which packets cross sectors, minimal loss of data in the recover process shown in Fig. 2.4 is essential. As such, the process starts by the discarding of the first two bytes. With these two bytes gone, the packet ID is attempted to be read again; as long as there is not a valid packet ID, bytes continue to be removed. With a valid packet ID the second half of recovery begins. It is not enough to use the discovered ID as the start of a valid packet, as IDs are not unique and can be packet data instead. To correct for this
problem, the length of the potential packet is used to extract the next potential ID. If this ID is in the valid set of IDs, the recovery process is complete. This methodology ensures that the loss from corrupted sectors is minimal because even if the recovery process flags an invalid packet as valid when it’s not, the process will start again with the next packet.

Verifying the sectors using CRC in Python presents difficulties due to CRCs being originally designed for hardware implementations. Without dedicated hardware, a naive algorithm is presented in [11]. The process assumes a CRC polynomial of length W, along with a shift register of the same size, which can be set to any desired seed value (typically 0xFFFF). The process starts by appending the message with 0’s of length W. The message is then shifted into the register, one bit at a time, with operations being performed based on the state of the popped bit. If the popped bit is zero, no further operations are performed and the next shift occurs. If the bit is one, the shift register is XORed with the polynomial. This process then continues until all bits of the message, including the augmented 0’s, have been shifted through. The shift register now contains the computed CRC for the message.

Importing a library capable of bitwise operations and coding the naive algorithm resulted in the correct CRCs, but also with a calculation time of over a second per sector. This is much too slow since even a relatively short collar run results in millions of sectors. With off-line processing taking too long, a speed up of the CRCs was desired. To achieve the desired speed up, one possible solution was to rewrite the algorithm using only native Python, with the hurdles that this included. This was not desirable due to Python’s lack of support for bit-wise operations resulting in several operations per bit. According to [11], it is also possible to implement CRCs using byte-wise methodologies. This is due to the property of XOR that states that regardless of the number of XORs performed on a register there will be a single XOR that has the same effect as all the others. This characteristic allows for the pre-computation of a table containing
Figure 2.4: Flowchart of the recovery process. When an invalid packet ID is found, either a whole sector has been marked as corrupt or data has been lost in some other form and it must be corrected. The process starts by the discarding of the first two bytes. With these two bytes gone, the packet ID is attempted to be read again; as long as there is not a valid packet ID, bytes continue to be removed. With a valid packet ID the second half of recovery begins. It is not enough to use the discovered ID as the start of a valid packet, as IDs are not unique and can be packet data instead. To correct for this problem, the length of the potential packet is used to extract the next potential ID. If this ID is in the valid set of IDs, the recovery process is complete.
all possible W-bit CRCs given an byte input. With this table the CRC can be calculated with fewer operations in byte increments. The byte-wise process starts by saving to a register the top byte of the shift register previously referenced in the naive implementation. One byte is then shifted into the register instead of one bit. Finally, the shift register is XORed with the pre-calculated CRC indexed by the previously saved top byte. The process continues in this byte-wise fashion until there are no bytes remaining. While currently the table is small enough to be computed at start up, due to the size of the CRC, this methodology would extend itself to larger CRCs with tables stored in external files.

Validation and verification techniques were used to ensure operation of the extractor. The data logging module was first modified to generate different headers and footers, testing that the extractor parsed the proper sectors correctly. With proper data being given, the CRCs were tested. Using first the naive implementation and then using the table-driven form, test CRCs were calculated on both the PC and microcontroller confirming that both generated identical values. Once CRCs were operational, data files were modified on the PC to force incorrect CRCs and to ensure that the discarding of sectors and the recovery process worked as required. With these steps complete, the extractor was tested on real data, confirming it was performing as expected.

Extracting and ensuring the data is correct is a vital step in the collar’s lifespan. With the need for cross-platform compatibility, Python, with its powerful capabilities, is the perfect language for implementation. Given the varied packets, external storage of IDs in a configuration file allows for easy growth and maintenance. Starting each file with ID information and packet details in a header allows extracted files to stand alone without other tools. Parsing the data in 512 byte sectors and discarding sectors without proper header and footers is the first step to ensuring data integrity. Taking the CRC after these steps ensures byte integrity of the entire sector and allows the packet dictionaries to be used to extract the data and flushing the data to a file. Recovery techniques ensure that,
in the event of data corruption, a minimal amount of data is lost. Validation and verification techniques were used to ensure that data collected by the collar would be recovered with a minimal loss in the event of data corruption.
2.7 Hardware Evolution

To facilitate the development of the mountain lion tracking collar, several sets of hardware were required. The initial foray into testing sensors was done on a breadboard controlled by the MicroChip Microstick. This dev kit required hardware modifications as in its base state the serial pinout could not supply power to the board. A simple solder jumper supplied power and significantly reduced the complexity of the development process. The sensors themselves (including the microSD card) were mounted on daughter boards from various vendors. Software was developed on this platform for hello world and to vet the hardware configuration.

With the hardware configuration verified, a smaller custom PCB was designed to allow for more extensive testing as shown in Fig. 2.5. This PCB leveraged the power of the ChipKit Uno32, previously used by the lab extensively, to bypass bringing up a new microprocessor. Due to the small size of the sensors in question there were issues with assembly that were mostly mitigated through the use of a modified laser-cut stencil procedure originally created by [13]. This pcb was used to both select the sensors used and to determine noise characteristics.

With the sensors selected, an even smaller PCB was designed to allow for limited deployment without ruggedization as shown in Fig. 2.6. This board has only the selected sensors along with accompanying microprocessor. Assembly issues continued within this variation until the introduction of kapton as a stencil material. The board is mainly single sided with the microSD card socket and and inductor for the power supply mounted on the bottom. This power supply proved unusable and an in-line power regulator was built into the cable connector was developed instead.

This iteration has been used since for all further testing and software development. Before ruggedization there are significant changes such as changing the processor and providing for slower clocks.
Figure 2.5: PCB Layout of the multi sensor development kit. The MEMs sensors and microSD card were powered by the UNO32 with the GPS having its own regulator to allow for 3.3V or 5V receivers. Note that the temperature sensor was not present in the original design, and was added manually after board assembly.
Figure 2.6: PCB layout of small scale collar tag. This board has only the selected sensors along with accompanying microprocessor. Assembly issues continued within this variation until the introduction of kapton as a stencil material. The board is mainly single sided with the microSD card socket and inductor for the power supply mounted on the bottom. This power supply proved unusable and an in-line power regulator was built into the cable connector was developed instead.
Chapter 3

Results

To simultaneously maximize performance while minimizing power consumption, the proper sensors must be chosen. Significant testing must be performed on the accelerometer and magnetometer to ensure that they are capable of performing the variable and dynamic sampling required to make the collar "power-aware." The use of both triaxial magnetometers and accelerometers allows for a simple dead reckoning navigation between GPS samples, which can be used to increase the information content delivered to the biologists. The trade-off is being made between accuracy of the sensor, time that it takes to wake from sleep to the first usable sample, and power draw (in both sleep and active modes). The power hungry nature of GPS receivers enforces a testing regimen that concentrates on startup performance in a variety of environmental conditions. Again, this is due to the fact that the bulk of the GPS receiver ON time is for acquisition, and then only a very short time for sampling before sleeping again.

As will be shown in the analysis of results, the down selection of various GPS receivers is driven by the difficulty in acquiring the satellite signals when in deep canyons under tree cover.

Once sensors are selected, power analysis allows for the prediction of collar lifespan. Using conservative estimates of power draw for all components gives a usable overall power budget and estimated lifespan. It will be shown that the
bulk of the power is consumed in the RF communications; hence compressing the data via on-board animal behavior identification is paramount to long collar life.

With the 3rd generation collar unavailable for deployment, biologists are able to use the 2nd generation of the collar to train models for predicting behavior using captive mountain lions. Applying these models to wild animal behavior allows for a proof of concept for dynamic sampling to be completed. In order to process the accelerometer data to identify animal behavior \textit{in situ} the accelerometer data is fed to a ”random forest” data analysis algorithm. The random forest has been trained using data from captive animals in realistic environments, and is then used to analyze data from 2nd generation collars. This results in being able to identify animal behaviors in post-processing [14]. While the random forest algorithm has not been tested on the embedded processor, once trained it does not require excessive computational capability and should be implemented on-board the collar (see Sec. 4.2).
<table>
<thead>
<tr>
<th>Sensor</th>
<th>Sample Rates</th>
<th>Scale</th>
<th>Output Bit Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freescale MMA8451Q</td>
<td>1.56Hz-800Hz</td>
<td>2,4,8g</td>
<td>14-bit</td>
</tr>
<tr>
<td>Bosch BMA180</td>
<td>1Hz-1200Hz</td>
<td>1,1.5,2,3,4,8,16g</td>
<td>14-bit</td>
</tr>
<tr>
<td>STmicro LSM303DLHC (accel)</td>
<td>1Hz-1344Hz</td>
<td>2,4,8,16g</td>
<td>16-bit</td>
</tr>
<tr>
<td>STmicro LSM303DLHC (mag)</td>
<td>1Hz-1344Hz</td>
<td>1.3-8.1 gauss</td>
<td>12-bit</td>
</tr>
<tr>
<td>Honeywell HMC5883L</td>
<td>0.75Hz-75Hz</td>
<td>1-8 gauss</td>
<td>12-bit</td>
</tr>
<tr>
<td>Freescale MAG3110</td>
<td>0.31Hz-80Hz</td>
<td>1000μT</td>
<td>16-bit</td>
</tr>
</tbody>
</table>

Table 3.1: Overview of accelerometers and magnetometers tested. The first three are triaxial accelerometers with varying sample times and dynamic range. The last three are triaxial magnetometers used for sensing Earth’s magnetic field (used for animal heading information). Note that the STmicro device in a combined accelerometer and magnetometer in a single device which is in some ways easier to integrate than two separate devices.

### 3.1 Accelerometer and Magnetometer

With the advent of the smartphone industry the availability of cheap low-power triaxial sensors (accelerometers and magnetometers for the collar) has increased tremendously. As such, a selection of sensors were selected for testing with the specifics models shown in Table 3.1.

These sensors were all configured to take data at as close to same sample rate and gain as possible with the microprocessor recording raw readings to the SD card. The development platform containing all the sensors was then held stable in various orientations while samples were taken. The raw values were then processed to equalize values to the same scale to create point clouds.

Scatter plots of overlayed accelerometer values during orientation trials are shown in Fig. 3.1. This overview already shows the differences between the accelerometers: in ideal conditions, all three sensors should contain identical point clouds. While the clouds are not identical, given the real world conditions, they are similar enough to indicate that all the sensors are working as expected from this scale. Further insights can be gained from manipulation of the individual clouds. Scatter plot of the Bosch accelerometer cloud during orientation trials are shown in Fig. 3.2. This zoomed plot shows that the Bosch accelerometer is
Figure 3.1: Scatter plots of overlayed accelerometer values during orientation trials. This overview already shows the differences between the accelerometers as: in ideal conditions, all three sensors should contain identical point clouds. While the clouds are not identical, given the real world conditions, they are similar enough to indicate that all the sensors are working as expected.
performing as expected with the denser clouds caused by the sensor being held in that particular orientation for longer periods of time. Scatter plot of the Freescale accelerometer cloud during orientation trials are shown in Fig. 3.3. This zoomed plot shows that the FreeScale accelerometer is performing as expected with results very similar to the Bosch. Scatter plot of the STMicro accelerometer cloud oriented to show banding issues are shown in Fig. 3.4; the banding indicates that the internal analog to digital conversion is at a lower precision than what is being used on the I2C communication bus. With the cloud oriented to align with a single plane, it is clearly showing that the STMicro has a quantization errors. While small sections seem to have higher resolution, overall the significant banding makes it unusable.

A similar scatter plot of overlayed magnetometer values during orientation trials are shown in Fig. 3.5. This overview already shows issues with both the Honeywell and STMicro being malformed in interesting ways. Both show groupings of points within sub-boxes with borders. Manipulation of zoom levels show the same behavior within each individual box. The reason that both sensors have similar odd behavior, not the reasoning behind the behavior, can by surmised by inspection of the datasheets which show the exact same register layout between the two magnetometers; the assumption is that in some form STMicro copied the silicon from Honeywell.

With the orientation trials complete, an accelerometer and magnetometer were selected. The obvious choice for the magnetometer was the Freescale, given the difficulties encountered by the Honeywell and STMicro. Quantization errors made the STMicro accelerometer unusable, leaving the Bosch and Freescale. While both showed similar performance, the Freescale is a newer device with greater resolution and an enhanced feature set which could prove beneficial in the field. With the Freescale magnetometer already being utilized, the Freescale Accelerometer was the obvious choice for use on the tracking collar.
Figure 3.2: Scatter plot of the Bosch accelerometer cloud during orientation trials. This zoomed plot shows that the Bosch accelerometer is performing as expected with the denser clouds caused by the sensor being held in that particular orientation for longer periods of time.
Figure 3.3: Scatter plot of the Freescale accelerometer cloud during orientation trials. This zoomed plot shows that the FreeScale accelerometer is performing as expected with results very similar to the Bosch.
Figure 3.4: Scatter plot of the STMicro accelerometer cloud oriented to show banding issues. With the cloud oriented to align with a single plane, it is clearly showing that the STMicro has a quantization error. While small sections seem to have higher resolution, overall the significant banding makes it unusable.
Figure 3.5: Scatter plot of overlayed magnetometer values during orientation trials. This overview already shows issues with both the Honeywell and STMicro being malformed in interesting ways. Both show groupings of points within sub-boxes with borders. Manipulation of zoom levels show the same behavior within each individual box. The reason that both sensors have similar odd behavior, not the reasoning behind the behavior, can by surmised by inspection of the datasheets which show the exact same register layout between the two magnetometers; the assumption is that in some form STMicro copied the silicon from Honeywell.
3.1.0.1 Sensor Stability

Sensor stability and noise performance are key parameters in hardware selection. Experiments were carried out to quantify the long term noise characteristics and stability of both triaxial sensors (the accelerometer and magnetometers) using techniques derived from atomic clock characterization. Many of these experiments were carried out by Sarah Dean, an undergraduate summer REU student under the author’s direction.

With the accelerometer and magnetometer selected, determining their noise characteristics was desired. To this end the model used is

\[ y_m = ky_t + b(t) + v_w \]

where \( y_m \) is the measured sensor output, \( y_t \) is the true physical quantity, \( k \) is a scale factor, \( v_w \) is a Gaussian white noise (zero mean, standard deviation \( \sigma_w \)), and \( b(t) \) is a time varying bias signal with \( b(t) \) changing with time and defined as

\[ \dot{b} = -\left(\frac{1}{\tau}\right)b + \omega \]

where \( \omega \) is normally distributed and \( \tau \) is a time constant.

An important state of this model occurs when the true signal is constant. If this is the case, all variations of the signal are the result of noise. Analyzing data in stable conditions allows the use of overbounding [15] to parameterize the noise.

In order to accomplish this, data was collected from the sensors for long durations in stable conditions (an unused lab on campus). Data was taken via microcontroller at 5Hz intervals while the data rate for the accelerometer was 6.25Hz and the magnetometer was set to 10Hz with an oversample ratio of 128. Magnetometer samples were observed to show effects from nearby electronics, so samples were clipped accordingly to relatively more stable portions.

The parameters of the noise were characterized using autocorrelation and an Allan variance analysis. The Allan variance was developed in the 1940’s to analyze
Table 3.2: Table of noise parameters for the accelerometer and magnetometer. These values are the averages taken from two different overnight trials. Note that the magnetometer noise characteristics are of limited use due to the temperature correlation.

<table>
<thead>
<tr>
<th></th>
<th>$\sigma_w$</th>
<th>$\sigma_b$</th>
<th>$\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>acc$_x$</td>
<td>0.6mg</td>
<td>0.1mg</td>
<td>98 s</td>
</tr>
<tr>
<td>acc$_y$</td>
<td>0.6mg</td>
<td>.1 mg</td>
<td>122 s</td>
</tr>
<tr>
<td>acc$_z$</td>
<td>0.7mg</td>
<td>0.1mg</td>
<td>122 s</td>
</tr>
<tr>
<td>mag$_x$</td>
<td>2.2$\mu$T</td>
<td>2.4$\mu$T</td>
<td>n/a</td>
</tr>
<tr>
<td>mag$_y$</td>
<td>2.5$\mu$T</td>
<td>2.4$\mu$</td>
<td>n/a</td>
</tr>
<tr>
<td>mag$_z$</td>
<td>4.3$mu$T</td>
<td>3.1$mu$T</td>
<td>n/a</td>
</tr>
</tbody>
</table>

The stability of atomic clocks and characterize the contributions of various noise sources (a complete treatment can be found in [16] and [17]. Fig. 3.6 shows an example of this technique for the X axis of the accelerometer. Fig 3.7 shows the same procedure being run on the magnetometer Y axis, and failing due to the noise of the magnetometer being correlated.

The magnetometer readings were correlated with temperature in a linear relationship as shown in Fig. 3.8. The temperature information was used to correct the magnetometer data and the overbounding method was performed again without success. This is most likely due to the noise from the temperature sensor itself. Further testing should be run in the future with a higher quality temperature sensor to fully characterize the noise. The parameter averages of the multiple trials are shown in Table 3.2. The results of the Allan variance analysis indicates that white noise dominates the sensors for averaging times of less than one minute. This sets an upper bound on any low-pass filter time constants used to extract the mean values of acceleration (which would indicate inclination of the animal track on a slope).

The temperature correlation of the magnetometer shows that measurements tend to be correlated, and thus averaging is of limited use. This could be mitigated with a better temperature correction method using better temperature sensors, though at a cost of increased power draw.
Figure 3.6: The Allan variance and autocorrelation plot generated for the X axis of the accelerometer. Note the nearly flat line of the autocorrelation plot indicating that the noise characteristics generated are correct.
Figure 3.7: The Allan variance and autocorrelation plot generated for the X axis of the magnetometer. This attempt at determining noise characteristics fails due to the magnetometer noise being correlated.
Figure 3.8: Scatter plot showing the noise correlation between magnetometer values and temperature. Temperature data was taken using an analog temperature sensor overnight to allow for temperature fluctuations in the testing environment. This plot clearly shows magnetometer values increasing with temperature.
### 3.1.0.2 Startup Times

During collar operation, various sensors will be power cycled to conserve power and maximize collar life. Therefore, it is very important to understand startup performance: the interval required from power on to valid data. This will also allow for accurate power estimation. Testing of startup times followed the following procedure:

- The sensor to test was turned off: via power mosfet or setting sensor to standby mode
- The sensor’s physical orientation was changed to ensure post-start readings would not be influenced by previous orientation (i.e. a memory effect).
- Sensor power was restored and data was taken at a rate higher than the output data rate of the sensor to ensure the start-up characteristics could be classified

Several trials were run using differing output data rates of the sensors for both power switching and sleeping.

Plotting the sensors’ output reveals an obvious startup error. To identify stability time in a consistent manner, a reference orientation was required. This was acquired by taking the distribution of the last 75% of the data. Since sensor stability was normally acquired within the first 2% of the data using the first point within 1.5 standard deviations of the stable orientation was deemed reliable.

The time taken to achieve the first stable point varied with the output data rate of the sensor. The number of sampling periods versus the output data rate is plotted in Fig. 3.9. In all cases it took at most five sampling periods to achieve stability, while most took less than three.

### 3.1.0.3 Power Consumption

To estimate collar life, characterization of sensor power consumption is required. As such, current draws of the accelerometer and magnetometer were
Figure 3.9: Plots of time to sensor stability in terms of output data rate for the magnetometer and accelerometer. From these plots we can see little gain from sleeping the accelerometer instead of powering it off. The magnetometer shows a slight improvement from sleep versus power off but with only three output data rates taking longer the difference is insignificant.

The current draws of the sensors were measured using a TI INA212 low side current shunt which has an internal gain of 1000. The sense resistor used was 1Ω and 4.7Ω respectively for the magnetometer and accelerometer. The current draws for both sensors were measured in both standby and active modes. Measurements in active modes were made across various output data rate settings.

As expected, the current draw of the sensors shown in Fig. 3.10 and Fig. 3.11 varied with the output data rate settings. Additionally, the sensors’ measured current were consistently 20μA higher than the data sheet’s values. This constant offset across all output data rate settings suggests a flaw in the measuring set up. In the case of the magnetometer there are settings for both output data rate and oversampling. The internal sampling rate are these two settings multiplied together, with this value being correlated with the current draw.

The average current draw were generally consistent with the data sheets; there
Figure 3.10: Plots showing current draw of the accelerometer in both active and standby modes with the red points being cited datasheet values. Current draw from the accelerometer increased with the output data rate as expected. There is a 20μA offset across all values which is assumed to be an error from the measuring setup.
Figure 3.11: Plots showing current draw of the magnetometer in both active and standby modes with the red points being cited datasheet values. Due to oversampling, the actual internal rate of the magnetometer is the output data rate * oversample ratio. With this accounted for, current draw increased with the internal data rate as expected. There is a 20μA offset across all values which is assumed to be an error from the measuring setup.
was variation in the current draw when the sensors were in active mode as shown in Fig. 3.12. Both the accelerometer and magnetometer displayed this pattern of skewed current distributions.

### 3.1.0.4 Power Budget

Given the very low power usage by the accelerometer and the magnetometer, there is to be gained by power cycling the sensors into sleep mode all the time. To estimate our power usage we use a static high sample rate to over bound our power usage. For instance, the algorithm used to identify animal behavior can use an input accelerometer rate of 32Hz. To guarantee samples of at least 32Hz and our 10:1 ratio, an accelerometer rate of 50Hz and a magnetometer rate of 5Hz is required. For the accelerometer this results in 24μA and total energy can be calculated as:

\[
P = V \times I = 3.6V \times 24\mu A = 86.4\mu W
\]  
\[
E = P \times t = 86.4mW \times 3600s = 311mJ
\]
For the magnetometer this 5Hz rate gives a current draw of $68.8\mu A$ and the energy can be calculated as follows:

\[
P = V * I = 3.6V * 68.8\mu A = 247.68\mu W \tag{3.3}
\]

\[
E = P * t = 247.68mW * 3600s = 891.64mJ \tag{3.4}
\]

Combining these two values gives a total energy usage of $1.203J$.

Typically, the batteries we would be using would have 13000mAmp-Hours at 3.6V, which would be run with at least three in parallel giving a total energy budget of:

\[
(3 * 13) * 3.6 * 3600 = 505.44KJ \tag{3.5}
\]

Using the total energy budget calculated of 505.44KJ, the MEMs sensors’ would consume 0.0002379% of the available budget, allowing sampling to continue for 47.94 years. Obviously there are other drains on the system besides a accelerometer and magnetometer, but it is important to benchmark all components for relative performance to indicate where to attack the endurance problem of the collar.
<table>
<thead>
<tr>
<th>Receiver</th>
<th>Channels</th>
<th>Current Draw while tracking</th>
<th>Max Update Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locosys LS20030</td>
<td>32</td>
<td>47mA</td>
<td>5Hz</td>
</tr>
<tr>
<td>MediaTek 3329</td>
<td>66</td>
<td>37mA</td>
<td>10Hz</td>
</tr>
<tr>
<td>San Jose FV-M8</td>
<td>32</td>
<td>42mA</td>
<td>5Hz</td>
</tr>
<tr>
<td>uBlox 5</td>
<td>50</td>
<td>Unknown</td>
<td>4Hz</td>
</tr>
<tr>
<td>SiRF Star EM-406</td>
<td>20</td>
<td>44mA</td>
<td>1Hz</td>
</tr>
</tbody>
</table>

Table 3.3: Summary of characteristics for GPS receivers tested.

### 3.2 GPS Selection and Power Budget

The GPS receiver is essential to the collar allowing biologists to track the animals. At the same time it has a high power drain while active and, as such, can not be powered most of the time. As power cycling is required to ensure long battery life, the characterization of startup performance is essential in choosing a GPS receiver. A selection of GPS receivers was chosen for these trials and are shown in Table 3.3. GPS receivers have three different types of starts:

- **Cold Start**: receiver has no information
- **Warm Start**: receiver has last position, ephemeris and current time
- **Hot Start**: receiver has all the information from the warm start plus all the satellites in view

To test the startup performance the rig for stationary trials was built. Each GPS unit was mounted to a board and provided its own USB-to-serial adapter. These adapters were connected to a laptop running a set of scripts in MATLAB. The scrips recorded the serial stream from each GPS and sent NMEA commands as needed to control the starts.

For baseline testing ideal conditions for the GPS receivers was chosen. The roof of the Jack Baskin Engineering building with its uncluttered view of the sky met these requirements. This location also provided power which allowed for start
tests to continue for multiple days. Due to power availability, each start was run for 24 hours with the start command being sent to the receiver every 5 minutes.

At the conclusion of the tests the rig was retrieved and the GPS serial stream files were parsed. GPS messages sent in the NMEA format contained several interesting pieces of information: acknowledgement of start, all devices acknowledged that a start command had been sent allowing a start time to be calculated; position and time messages, give the GPS receivers currently calculated position; and fix, a marker indicating the accuracy of the position according to the GPS receiver itself. The GPS fix, with possible values of no fix, 2D fix and 3D fix, is the only way of determining accuracy without an external source. As such, while on the collar, the position given with the first fix is the main data point of interest.

These first fixes were saved and then converted to a North-East-Down (NED) coordinate frame using the average of all valid fixes as the zero location. The points were then plotted, along with their $1-\sigma$ ellipses, as shown in Fig. 3.13, 3.14, and 3.15 to compare relative accuracies of the GPS receivers.

Scatter plot showing cold start position accuracies under ideal conditions is shown in Fig. 3.13. In ideal conditions, it is very clear that the uBlox GPS receiver has the greatest accuracy, with the Locosys and MediaTek following behind. Note that for the tracking collar application, it is expected that the GPS receiver will rarely be operating in ideal conditions. Scatter plot showing warm start position accuracies under ideal conditions is shown in Fig. 3.14. In ideal conditions, during a warm start, the uBlox GPS receiver has the greatest accuracy, with the Locosys and MediaTek receivers following behind but closer than the cold start. During normal collar operation, it is expected that warm starts will be the most common start experienced although, again, not under ideal conditions. Scatter plot showing hot start position accuracies under ideal conditions is shown in Fig. 3.15. In ideal conditions, it is very clear that the uBlox GPS receiver still the greatest accuracy, with the Locosys and MediaTek following behind. During collar operation, hot starts will rarely be encountered due to the power drain from the
Figure 3.13: Scatter plot showing cold start position accuracies under ideal conditions. Each receiver was forced to do a hard restart with only the factory ephemeris data enabled, showing the position accuracy after the first 2D fix. In ideal conditions, it is very clear that the uBlox GPS receiver has the greatest accuracy, with the Locosys and MediaTek following behind. Note that for the tracking collar application, it is expected that the GPS receiver will rarely be operating in ideal conditions.
Figure 3.14: Scatter plot showing warm start position accuracies under ideal conditions. Each receiver was forced to do a restart with last position, ephemeris, and current time, showing the position accuracy after the first 2D fix. In ideal conditions, during a warm start, the uBlox GPS receiver has the greatest accuracy, with the Locosys and Mediatek receivers following behind but closer than the cold start. During normal collar operation, it is expected that warm starts will be the most common start experienced although, again, not under ideal conditions.
Figure 3.15: Scatter plot showing hot start position accuracies under ideal conditions. Each receiver was forced to do a restart with last position, ephemeris, current time, and satellites in view, showing the position accuracy after the first 2D fix. In ideal conditions, it is very clear that the uBlox GPS receiver has the greatest accuracy, with the Locosys and MediaTek following behind. During collar operation, hot starts will rarely be encountered due to the power drain from the GPS.
GPS; the only time this would not be true would be for segments of high fidelity continuous sampling.

Time to lock is the primary way to conserve power because it reduces active time for the receiver. Accurately measuring time to lock is a complicated process due to the characteristics of the GPS receivers. As some of the starts entail not having accurate time at start, the time to lock can not be determined by the parsing of messages directly. Instead they can be inferred from the number of messages that the GPS has transmitted. Location messages from all of the receivers were subscribed to at 1Hz intervals; this sampling rate was the fastest at which all receivers could output. This calculated time to fix versus variance of location is shown in Fig. 3.16, 3.17, 3.18.

Scatter plot of cold start variance versus time to first 2D fix under ideal conditions is shown in Fig. 3.16. In ideal conditions, the uBlox GPS receiver has the greatest number of fast, low variance starts with the the MediaTek and Locosys following close behind. Scatter plot of warm start variance versus time to first 2D fix under ideal conditions is shown in Fig. 3.17. In ideal conditions, the Ublox and Locosys GPS receivers have low variance positions but several outliers where fixes take substantially longer. In contrast, the MediaTek and San Jose Receivers have a higher variance but no outliers with long fix times. Scatter plot of hot start variance versus time to first 2D fix under ideal conditions is shown in Fig. 3.18. In ideal conditions, all receivers tested had tight clustering below 3 seconds except the San Jose which had most of its locks concentrated over the 3 to 4 second interval.

In ideal conditions, the uBlox performs best with fast lock times and low variance.

In addition to these plots, other plots were generated from the start data. Histograms of GPS height at first 2D fix after cold start under ideal conditions are shown in Fig. 3.19. All receivers clustered together around true height with the exception of a single spike left of true height; note that the GPS uBlox receiver’s
Figure 3.16: Scatter plot of cold start variance versus time to first 2D fix under ideal conditions. In ideal conditions, the uBlox GPS receiver has the greatest number of fast, low variance starts with the MediaTek and Locosys following close behind.
Figure 3.17: Scatter plot of warm start variance versus time to first 2D fix under ideal conditions. In ideal conditions, the Ublox and Locosys GPS receivers have low variance positions but several outliers where fixes take substantially longer. In contrast, the MediaTek and San Jose Receivers have a higher variance but no outliers with long fix times.
Figure 3.18: Scatter plot of hot start variance versus time to first 2D fix under ideal conditions. In ideal conditions, all receivers tested had tight clustering below 3 seconds except the San Jose which had most of its locks concentrated over the 3 to 4 second interval.
spike is substantially smaller than the others. As the collar is power-limited, time to achieve a 3D fix is not available, and thus a 2D fix's height is of limited use. Scatter plots of GPS position after cold start, showing how each GPS walked in its location until the trial timed out are shown in Fig. 3.20. It is clear that both the San Jose and Locosys wandered around the first point found, while MediaTek and uBlox corrected to near true and refined estimates from there. While this can be used to infer the number of samples required for an accurate fix, there is little point in performing this test for the other starts.

As can be seen in Fig. 3.21-3.26 and Fig. 3.27-3.32, the same experiments that we performed in ideal conditions for GPS (clear view of the sky, no blockages) are replicated for both woods and canyon environments. The "woods" test is under deep redwood forest canopy on the UCSC campus, and the "canyon" test was in a deep ravine close to the Jack Baskin Engineering building. The canyon walls form an approximately 40º deep "V."

Both the tree cover and steep walls of the canyon make GPS tracking difficult as the signals from the satellites are attenuated by both the foliage and the ground. Additionally, multi-path effects (no direct line of sight to satellites) corrupt the position and make it difficult for the GPS receiver to lock onto its correct position or even obtain any position fix at all. Due to the tough terrain, instead of taking data for set times, the GPS units were reset as soon as all units had sent a fix or a global timeout had occurred. This timeout was set to 5 minutes with the assumption that longer without a fix would be too large of a drain on the batteries in the collar. Additionally, as there was no AC power, the test were limited to the battery life of the PC which severely reduced the testing time.

The woods and canyon tests are more representative of environments that the Mountain Lions inhabit, and are quite difficult for the GPS to obtain decent position information. As such, we use these tests to drive the selection of our GPS receiver, since quick time to fix is paramount to obtaining data on interesting events. Note here that for different terrestrial animals, a different selection criteria
Figure 3.19: Histograms of GPS height at first 2D fix after cold start under ideal conditions. All receivers clustered together around true height with the exception of a single spike left of true height; note that the GPS uBlox receiver’s spike is substantially smaller than the others. As the collar is power-limited, time to achieve a 3D fix is not available, and thus a 2D fix’s height is of limited use.
Figure 3.20: Scatter plots of GPS position after cold start, showing how each GPS walked in its location until the trial timed out. It is clear that both the San Jose and Locosys wandered around the first point found, while MediaTek and uBlox corrected to near true and refined estimates from there. While this can be used to infer the number of samples required for an accurate fix, there is little point in performing this test for the other starts.
would make more sense. The collar is, however, designed with flexibility in mind and can accommodate different GPS receivers relatively easily.

Scatter plot showing cold start position accuracies under woods conditions after the first 2D fix is shown in Fig. 3.21. This plot very clearly shows the Ublox GPS Receiver having the greatest accuracy with the rest following the same order as previous trials. Note that the Sirf Star receiver has better accuracy than the MediaTek but its results are more skewed along a single axis. During this test the receivers were operating much closer to collar conditions, and as such, the woods tests had a much greater weight in the selection process than ideal conditions. Scatter Plot showing hot start position accuracies under woods conditions after the first 2D fix is shown in Fig. 3.22. This plot clearly shows that the San Jose has much worse position accuracy compared to the other receivers with fifty less fixes. The other receivers all achieved the same number of fixes with the Ublox attaining slightly higher accuracy than the others. Scatter Plot showing hot start position accuracies under woods conditions after the first 2D fix is shown in Fig. 3.23. This plot clearly shows that the San Jose has much worse position accuracy compared to the other receivers with fifty less fixes. The other receivers all achieved the same number of fixes with the Ublox attaining slightly higher accuracy than the others.

Scatter Plot of cold start variance versus time to lock under woods conditions after the first 2D fix are shown in Fig. 3.24. This plot shows that all the receivers had difficulty in achieving accurate fixes under woods conditions. All receivers tested had significant outliers although the San Jose and Sirf Star had the most. Scatter Plot of warm start variance versus time to lock under woods conditions after the first 2D fix are shown in Fig. 3.25. The Sirf Star and San Jose continue to have issues in outliers while the others continue the trend from the cold start with no receiver distinguishing itself from the others. Scatter Plot of hot start variance versus time to lock under woods conditions after the first 2D fix are shown in Fig. 3.26. This plot clearly shows that the majority of fixes occurred within 30
Figure 3.21: Scatter plot showing cold start position accuracies under woods conditions after the first 2D fix. This plot very clearly shows the Ublox GPS Receiver having the greatest accuracy with the rest following the same order as previous trials. Note that the Sirf Star receiver has better accuracy then the MediaTek but its results are more skewed along a single axis. During this test the receivers were operating much closer to collar conditions, and as such, the woods tests had a much greater weight in the selection process than ideal conditions.
Figure 3.22: Scatter plot showing warm start position accuracies under woods conditions after the first 2D fix. This plot very clearly shows that the Ublox GPS Receiver continued to have superior performance although it fixed its positions fewer times than the MediaTek or Locosys. The Locosys and MediaTek’s high number of fixes is offset by their low accuracy while the San Jose performed comparably to the other receivers.
Figure 3.23: Scatter Plot showing hot start position accuracies under woods conditions after the first 2D fix. This plot clearly shows that the San Jose has much worse position accuracy compared to the other receivers with fifty less fixes. The other receivers all achieved the same number of fixes with the Ublox attaining slightly higher accuracy than the others.
Figure 3.24: Scatter Plot of cold start variance versus time to lock under woods conditions after the first 2D fix. This plot shows that all the receivers had difficulty in achieving accurate fixes under woods conditions. All receivers tested had significant outliers although the San Jose and Sirf Star had the most.
Figure 3.25: Scatter Plot of warm start variance versus time to lock under woods conditions after the first 2D fix. The Sirf Star and San Jose continue to have issues in outliers while the others continue the trend from the cold start with no receiver distinguishing itself from the others.
Figure 3.26: Scatter Plot of hot start variance versus time to lock under woods conditions after the first 2D fix. This plot clearly shows that the majority of fixes occurred within 30 seconds across all receivers. The MediaTek and Ublox both have fast fixes with some slower outliers while the Locosys has fewer outliers but a longer minimum time to fix. The Sirf Star outperformed all the others in achieving fixes with no single fix taking more than 10 seconds to achieve.
seconds across all receivers. The MediaTek and Ublox both have fast fixes with some slower outliers while the Locosys has fewer outliers but a longer minimum time to fix. The Sirf Star outperformed all the others in achieving fixes with no single fix taking more than 10 seconds to achieve.

Scatter plot showing cold start position accuracies under canyon conditions after the first 2D fix as shown in Fig. 3.27. It is very clear from this plot that the Ublox and Sirf star are not up to the task with fewer than half the fixes of either the Mediatek or the Locosys. The San Jose actually performs better in this case but is still very inaccurate. The Locosys has the most fixes and the tightest circle but has an interesting offset with most fixes to the left of true. Note that the low performers, Sirf Star and Ublox, have the tightest circles of all receivers; it is assumed that the firmware onboard those receivers require greater confidence before they will pronounce a fix. Scatter plot showing warm start position accuracies under canyon conditions after the first 2D fix as shown in Fig. 3.28. Unlike the previous cold start trial the warm starts mimic the woods and ideal conditions with the uBlox having the highest accuracy, albeit with 20 fewer fixes than any other receiver. The MediaTek came closest in accuracy to the uBlox but with fewer fixes than either the Locosys or the Sirf Star. Scatter plot showing hot start position accuracies under canyon conditions after the first 2D fix as shown in Fig. 3.29. With the additional information available to the receivers, this trial is very similar to trials performed in ideal conditions except for the San Jose which performed far worse in this trial.

Scatter Plot of cold start variance versus time to lock under canyon conditions after the first 2D fix as shown in Fig. 3.30. This plot shows that all receivers have substantially decreased accuracy of their fixes along with increased times to achieve fix. Of interest is the cluster of fixes from Locosys which fix immediately upon start; this is the same cluster of fixes that have the observed offset in the position plot. It is unknown what causes the Locosys to exhibit this behavior which does not occur in the other trials.
Figure 3.27: Scatter plot showing cold start position accuracies under canyon conditions after the first 2D fix. It is very clear from this plot that the Ublox and Sirf star are not up to the task with fewer than half the fixes of either the Mediatek or the Locosys. The San Jose actually performs better in this case but is still very inaccurate. The Locosys has the most fixes and the tightest circle but has an interesting offset with most fixes to the left of true. Note that the low performers, Sirf Star and Ublox, have the tightest circles of all receivers; it is assumed that the firmware onboard those receivers require greater confidence before they will pronounce a fix.
Figure 3.28: Scatter plot showing warm start position accuracies under canyon conditions after the first 2D fix. Unlike the previous cold start trial the warm starts mimic the woods and ideal conditions with the uBlox having the highest accuracy, albeit with 20 fewer fixes than any other receiver. The MediaTek came closest in accuracy to the uBlox but with fewer fixes than either the Locosys or the Sirf Star.
Figure 3.29: Scatter plot showing hot start position accuracies under canyon conditions after the first 2D fix. With the additional information available to the receivers, this trial is very similar to trials performed in ideal conditions except for the San Jose which performed far worse in this trial.
Figure 3.30: Scatter Plot of cold start variance versus time to lock under canyon conditions after the first 2D fix. This plot shows that all receivers have substantially decreased accuracy of their fixes along with increased times to achieve fix. Of interest is the cluster of fixes from Locosys which fix immediately upon start; this is the same cluster of fixes that have the observed offset in the position plot. It is unknown what causes the Locosys to exhibit this behavior which does not occur in the other trials.
Figure 3.31: Scatter plot of variance versus time to lock under canyon conditions after the first 2D fix. With the increased information given by the warm start this plot shows that the GPS receivers are able to achieve fixes with increased time and reduced accuracy. All receivers had significant outliers with MediaTek having the fewest.
Figure 3.32: Scatter plot of variance versus time to lock under canyon conditions after the first 2D fix. This plot shows that the increased information given from the hot start are generally able to acquire fixes under ten seconds with decreased accuracy barring the very poor performance of the San Jose.
lock under canyon conditions after the first 2D fix as shown in Fig. 3.31. With the increased information given by the warm start this plot shows that the GPS receivers are able to achieve fixes with increased time and reduced accuracy. All receivers had significant outliers with MediaTek having the fewest. Scatter plot of variance versus time to lock under canyon conditions after the first 2D fix as shown in Fig. 3.32. This plot shows that the increased information given from the hot start are generally able to acquire fixes under ten seconds with decreased accuracy barring the very poor performance of the San Jose.

Over the course of the start trials Ublox consistently outperformed the other receivers until the canyon trials. With this trial both the uBlox and Sirf Star were no longer contenders due to their inability to get fixes within reasonable time frames. With the San Jose consistently performing the worst across all trials, this leaves only the MediaTek and the Locosys. Throughout the majority of the trials the Locosys slightly outperformed the MediaTek which is interesting given that they both use the same chipset. In canyon trials an anomaly with offset starts and instant fixes made the Locosys unusable. With its decent accuracy and lock times across all conditions along with a small and easy to interface form factor made MediaTek the correct choice for the tracking collar.

3.2.0.5 Power Usage

The advantage of the ANIMA collar is that it reduces the power used when it can by using lower sampling rates and turning off various sensor subsystems. Conversely, when interesting events are detected, it uses much greater power to collect sensor data at a much higher temporal fidelity.

In order to estimate the amount of power used by the GPS, we use a simple duty cycle argument to conservatively overbound our power usage. For instance, the GPS receiver uses approximately 30mA at 3.3V while acquiring satellites. That is, 99mW of power, and takes at a worst case 55 seconds to lock on to the signals. With our standard 3 samples spaced at 1 second apart followed by
sleeping the GPS receiver for the remainder of the sample time (e.g.: one hour), we can calculate the total energy used as:

\[ P = V \times I = 3.6V \times 30mA = 108mW \]  \hspace{1cm} (3.6)

\[ E = P \times t = 99mW \times (55 + 3s) = 6.264J \]  \hspace{1cm} (3.7)

Typically, the batteries we would be using would have 13000m\text{Amp-Hours} at 3.6V, which would be run at least 3 in parallel giving a total energy budget of:

\[ (3 \times 13) \times 3.6 \times 3600 = 505.44KJ \]  \hspace{1cm} (3.8)

Meaning that the GPS, sampled normally once per hour, would consume 0.0012393\% of the available energy, conversely allowing the GPS only to run once per hour for 9.2 years. Obviously, there are other drains on the system other than just the GPS, however it is important to benchmark the power consumption for reliable estimates of collar life.

3.3 Data Logging
3.4 Data Logging Power Budget

The results of data logging have already been covered in Section 2.5 in the Verification and Validation comments; the use of a microSD card for archival storage has the advantages of robust, inexpensive bulk storage that is independently accessible on a PC. Data integrity is paramount, and several features have been added to ensure that valid data can be recovered even in the event of corruption of the card. The microSD card remains a fairly big power draw on the collar, and as such its use should be minimized where possible (see Section 4.2 for some strategies to do so).

In order to estimate the amount of power used by Data logging, we use an average sampling frequency based on a high data rate on the primary sensors. While this is not entirely realistic, it is sufficiently conservative to give us upper bounds on power consumption.

Using the same rates as described in Section 3.1 and Section 3.2 we can determine the number of bytes used per hour as shown below: From the total bytes written of 1224000 is translated to 2428.62 sectors per hour. From Ref. [18] datasheet for a microSD card it pulls 100mA in slow speed mode while writing and 250uA while sleeping. Tests performed in the lab give a total duration of each sector write of 6.46ms including the 5ms of inactivity before the card sleeps. The time taken for Sector Writes are calculated as follows:

\[
T_a = \frac{\text{Total Bytes}}{\text{Bytes per Sector}} \times t = \frac{1224000}{504} \times 6.46 ms = 15.6913s \quad (3.9)
\]

\[
T_s = 1h - T_a = 60 \times 60 - 18.291s = 3584.3086s \quad (3.10)
\]
As such we can calculate total energy usage as follows:

\[
P_a = V \times I = 3.6V \times 100mA = .36W \tag{3.11}
\]
\[
E_a = P \times t = .36W \times 15.6913s = 5.6489J \tag{3.12}
\]
\[
P_s = V \times I = 3.6V \times 250\mu A = .9mW \tag{3.13}
\]
\[
E_s = P \times t = .9mW \times 3584.3086s = 3.2258J \tag{3.14}
\]
\[
E_t = E_a + E_s = 5.6489J + 3.2258J = 8.874760J \tag{3.15}
\]

Typically, the batteries we would be using would have 13000mAmp-Hours at 3.6V, which would be run at least 3 in parallel (meaning a total energy budget of):

\[
(3 \times 13) \times 3.6 \times 3600 = 505.44KJ \tag{3.16}
\]

Using the total energy budget of 505.44KJ we can calculate that at this sampling rate the collar would consume 0.0017558% of the available energy, conversely allowing data logging to run for 6.49 years. While there will be periods of higher energy due to the need for high-fidelity data there will also be long periods significantly slower sampling which allows this to be an accurate estimate of energy usage on the collar.
### Table 3.5: Table of Subsystem usage. Summary of values determined in previous sections along with the total subsystem power usage.

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Energy Use (hourly)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>311mJ</td>
</tr>
<tr>
<td>Magnetometer</td>
<td>891.64mJ</td>
</tr>
<tr>
<td>GPS</td>
<td>6.264J</td>
</tr>
<tr>
<td>Data Logging</td>
<td>8.874760J</td>
</tr>
<tr>
<td>Total</td>
<td>13.963J</td>
</tr>
</tbody>
</table>

#### 3.5 Overall Power Budget

With the subsystems' power budgets estimated in 3.1, 3.2 and 3.4 shown in Table 3.5, power estimates for the whole system can be calculated. To do so, power estimates for the microprocessor must first be calculated. This is done for three configurations: the current board using the high-power microprocessor; a revised board using a low-power microprocessor; and the revised board with the low-power processor entering sleep state in a naive fashion.

The development board currently in use has a PIC32MX250F128D using the internal FRC and PLL to achieve a 40Mips clock. From the datasheet [19], max current draw is 30mA at that clock speed. At 3.6V we can calculate the energy used as:

\[
P = V \times I = 3.6V \times 30mA = 108mW
\]

\[
E = P \times t = 108mW \times 3600s = 388.8J
\]

With processor energy usage of 388.8J, there is a total energy usage of 402.764. Typically, the batteries we would be using would have 13000mAmp-Hours at 3.6V, which would be run at least three in parallel giving a total energy budget of:

\[
(3 \times 13) \times 3.6 \times 3600 = 505.44KJ
\]
Giving a total collar usage of 402.7636678J. This means that the entire collar would use 0.0796858% of the batteries per hour, allowing the collar to run for 52.288 days. This is expected given that it is a high-end processor running at maximum clock speed.

If we instead exchange the PIC32 for the PIC24F16KA102, the max current draw is reduced to 18mA [20]. At 3.6V the energy used is:

\[
P = V \times I = 3.6V \times 18mA = 64.8mW
\]
\[
E = P \times t = 64.8mW \times 3600s = 233.28J
\]

Giving a total collar usage of 247.2436678J. Using the previous energy budget of 505.4KJ, the collar would now use 0.0489165%, allowing the collar to run for 85.179 days.

The estimates made so far operate under the assumption that the microprocessor does not sleep. If the processor can sleep between samples, significant power can be gained. For this calculation we assume that a MEMs sensor takes 1 millisecond to sample (likely an order of magnitude longer given the 400Khz clock) as shown in Table 3.6. This table also shows the total duration of each sensor using the sample rates previously described. Assuming that all these samples occur independently, the power used by the microprocessor can be calculated assuming it is able to sleep for 50% of the time available:
<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Frequency</th>
<th>Time spent per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>50Hz</td>
<td>.05s</td>
</tr>
<tr>
<td>Magnetometer</td>
<td>5Hz</td>
<td>.005s</td>
</tr>
<tr>
<td>GPS</td>
<td>0.000277778Hz</td>
<td>0.016111111</td>
</tr>
<tr>
<td>Data Logging</td>
<td>0.674616953Hz</td>
<td>0.000984941</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>0.072096052s</strong></td>
</tr>
</tbody>
</table>

Table 3.6: Table summarizing sensors active time and the total time the processor needs to be active. Even assuming that sensors do not overlap, the microprocessor spends the vast majority of its time waiting. Note that the accelerometer and magnetometer uses an estimate of 1ms per I2C transmission of a 6-byte triplet.

\[ t_a = \frac{1}{t_{\text{per second}} \times 50\%} = \frac{0.072096052}{0.5} = 0.144192104 \]  
\[ t_s = 1 - t_a = 0.855807896s \]  
\[ P_s = v \times i = 3.6v \times .98\mu A = 3.5280\mu W \]  
\[ E_a = P \times t = 64.8mW \times 3600 \times 0.144192104s = 3.364J \]  
\[ E_s = P \times t = 3.5280\mu W \times 3600 \times 0.855807896s = 0.010869445J \]  
\[ E_t = E_s + E_t = 3.374582841J \]  

Giving a total collar usage of 17.338J. Using the previous energy budget of 505.4KJ, the collar would now use 0.0034303%, allowing the collar to run for 1214.65 days.

This power analysis shows that for the high-power processor, lifespan is less than two months. The switch to the low power processor only extends the life to three months. Sleeping the processor allows the lifespan to be extended to over 3 years. With an energy usage of less than .1W for the entire collar, further gains are limited. Conversely, the radio link with its 30W transmitter, can produce substantial gains with only a slight reduction in usage.

Indeed, the UHF/VHF radio link is far away the most power hungry device in the entire collar system. Using the lower power processor with native sleep
modes (triggered by an external Real Time Clock [RTCC]) gives us over three year deployments *ignoring the RF transmitter*. This indicated where progress can be made in making the system power aware: reducing the data and frequency of transmissions required will directly extend the life of the collar. Encoding animal behavior into a simple symbol table along with position can be used to reduce the collar radio payload.

Additionally, the radio can be set in an "answer only" mode where the collar will not turn on the transmitter unless it first receives a handshake from the biologists’ radio system. This would enable very low duty cycles on the radio, and would obviate the problem of spending time on transmission when no one is listening.
3.6 Biologist Behavior Analysis

Ultimately, the test of success with the mountain lion tracking collars rests in its ability to identify animal behavior in situ from the on-board sensors. While the scope of this thesis is largely the creation of the technology to enable this identification, and not the algorithms to do so explicitly, nonetheless this thesis would be incomplete without a discussion of the experimental results thus derived. What follows is largely a recreation of the work done in [21], and the reader is referred to that PhD thesis for a complete discussion of the results.

The biologists’ goal was to determine if it was possible to identify behavior on the basis of accelerometry data. To achieve this, 2nd generation collars were used to train Random Forest classification models based on captive animal behavior and to apply these models to wild animals.

From 2010-2011, 12 mountain lions were outfitted with the second generation collar. Originally the accelerometers were programmed to record data in a duty-cycle of two weeks on and four weeks off. While accelerometers were active they were sampled at 64Hz. In addition to increased accelerometer sample rates, the GPS was sampled at 5-minute intervals for one hour per day for a week within the active sampling period. After April 2011, this sampling regimen was changed to sample for two consecutive days every week. With the increased duration of sampling of accelerometers, the GPS regimen was altered to sample at 15-minute intervals for 24 hours. All collars were retrieved but, due to various failures, six data sets were unaccessible. From the viable collars, 4-26 days of usable data was extracted.

To train the Random Forest model, two captive mountain lions were outfitted with tracking collars and video recorded performing various behaviors. 1-2 trials were done per animal over two different visits. The video recordings were reviewed and categorized into mobile and non-mobile behaviors. Behavior data was divided into two second segments due to the length of the test track. This resulted in 2142 discrete behavioral observations, including walking, grooming, resting and
Random Forests (RF) [22] were chosen as the modeling tool due to their powerful machine learning capabilities for non-linear systems. The first model split mobile and non-mobile behavior. Random Forests are generated by creating a series of decision trees without duplicating input data. The Random Forest is the mode of the outputs of all the decision trees used. For determining the mobile behavior trees of size 500 were used. This procedure was repeated with the same parameters to determine the five classes of behaviors.

The RF generated from this data was then used to classify wild animal behavior. The mobility model correctly classified movements 96.17% of the time. Results from a regression model showed that model prediction of mobility were highly correlated with the distance traveled ($\beta = 5.245$, standard error = 0.174, $p < .001$). The behavior model predicted resting, low and high acceleration movement with 96.8%, 93.8%, and 92% accuracy, respectively. The model fared less well with feed behavior at 65.7% accuracy while grooming activity was not detected at all. This is largely due to the fact that grooming behavior does not involve much motion, hence there is little signal for the accelerometer to report.

The recorded data was then down-sampled from 64Hz to 32, 16, 8, 4, and 2Hz. Using the down-sampled sensor data, RF models were generated following the same procedure as before. These slower sample rate models identified behavior without significantly reducing accuracy until below 8Hz.

The biologists’ work has proven that dynamically sampling the sensors can work. With the Random Forest parameters loaded onto the collar, in situ behavior analysis should be possible. Furthermore, the insignificant accuracy loss from lowering the sample rate should reduce power usage by only requiring high sample rates during interesting behaviors.
Chapter 4

Conclusion and Future Work

4.1 Conclusions

This thesis covers the design and implementation of a power-aware dynamically sampling wildlife collar. It primarily concerns itself with the testing and selection of sensors along with custom hardware and software to implement the requirements of the collar.

Accelerometer and magnetometer samples are an integral part of the collar allowing for behavioral analysis as well as allowing for rudimentary dead reckoning to augment GPS positioning. Communication over I2C allows for easy sensor additions while a generic I2C driver provides a robust base on which to build specific sensor drivers.

A GPS receiver is required to allowed for animal tracking necessitating the need for a serial link and NMEA message parsing. Given the receiver’s inability to sample on demand, and the GPS receiver’s high power consumption, a sampling state machine was implemented. This state machine allows the receiver to be toggled between continuous and interval sampling on demand.

With multiple sensors sampling at different rates, a sampling architecture was developed to integrate all the sensors. With sensors being able to be sampled at arbitrary rates (at both hourly or millisecond scales), dynamic sampling is pos-
sible, although hardware timer limitations forced the use of a single base timer. With complementary sampling rates found for the accelerometer and magnetometer in 10:1 ratios, this module allows for their samples and others to be passed on to different software modules as needed.

Data requiring archiving comes in a wide variety of lengths due to the realities of the various sensors and different sampling rates. To ensure efficient packing on the storage medium, they are split across 512-byte sectors as needed. Minimal data corruption is essential, resulting in significant data integrity precautions including headers, footers, and CRCs. CRCs are generated using DMA to ensure low overhead and to allow sampling to continue while transferring data to the microSD card is in progress.

Data Logging allows for sectors to be written to the microSD Card with minimal overhead. With initialization routines designed to ensure the filesystem remains intact over the lifespan of the collar, the data files can be read by any PC. Using a standard library from MicroChip as a base, the sector write functions were rewritten to a polled hierarchical state machine for handshaking with DMA being used for the actual data transmission.

Data extraction allows for data files to be extracted on any PC running Python. Output files are prefixed with field identifying information to allow data processing to occur without access to the extractor. Checking the integrity of the file, including a table-driven CRC implementation to significantly speed up extraction, along with a recovery process for when sectors fail, allows for robust recovery in the event of data corruption.

Several iterations of hardware were required for the development process. Starting with bread-boarded circuits to allow initial drivers to be written, the hardware evolved into a custom PCB designed to test all MEMs sensors, at the same time leveraging a familiar processor to minimize development time. With sensor selection complete, a small scale version was designed and implemented to allow for continued development and non-ruggedized deployment. Assembly
issues of the small scale version resulted in a malfunctioning power supply but a
daughter power board alleviated this problem with minimal changes.

With a selection of sensors chosen, orientation trials were conducted as the
first step in choosing a magnetometer and accelerometer. These trials showed
that only one magnetometer performed as desired, and the Freescale MAG3110
magnetometer was chosen. With similar performance between the Bosch and
Freescale accelerometer, the Freescale MMA8451Q with its increased resolution
and feature set was chosen. These sensors’ noise characteristics were determined
by Sarah Dean, a REU summer student, and it was discovered that the magnet-
tometer is correlated with temperature. Additionally, power testing showed that
the small power draw of the sensors allowed them to be sampled continuously for
multiple years, given the assumed power budget.

The power-hungry nature of GPS receivers drove startup performance to be the
key criteria in selecting a receiver. Receivers were tested over cold, warm, and hot
starts in a variety of environmental conditions including a steep canyon. While
the uBlox receiver performed spectacularly over most of the trials, the dismal
performance in the canyon cold start trial led to the MediaTek 3329 receiver being
chosen. Power analysis using the sampling state machine developed in Section 2.2
showed that the GPS could be sampled hourly for multiple years.

Using sample rates determined in Section 3.1 and 3.2, allowed power budget
analysis for the entire collar. Repeating this process with different processor
variants showed substantial power gains are available by switching to a PIC24 low
power processor and enabling the native sleep modes.

While collar deployment was originally planned, delays in the development
process have not allowed for it to occur. The biologists, using the 2nd gener-
ation collar, have shown that behavior can be inferred from accelerometer data
given adequate sample rates using a Random Forest classification tool trained on
experimental data derived from captive animal studies.

At this point in the project, observations about the power budget can be made.
Given the energy budget calculated, collars should last for several years barring increases in microSD card capacity, yet current collars only last a few months. This is most likely due to the immense power drain from the radio system. As such, the most immediate power savings come from minimizing radio uptime. This should be a key focus of future work.

4.2 Future Work

4.2.1 Minimizing Radio Uptime

With the power budget dominated by the radio, significant changes should be made to the radio architecture. Given behavioral data allowing for adaptive sampling, there is no reason why similar changes could not be made in reporting GPS locations. The radio currently reports locations on set intervals every few hours; there is no reason these reports could not be made dynamically with behavior information included. Additionally, the radio receiver using significantly less power than transmission, and GPS providing accurate time, allows for very short synchronized checkups; if there is a need, the collar can then turn on its transmitter and respond as necessary. Additional gains could be made from implementing compression on board the microprocessor, as even a long compression process onboard uses far less energy than a radio transmission.

4.2.2 Low Power Processor

To ease the design process, a high-power processor with many advanced peripherals was used for development work. This processor’s powerful capabilities, along with plenty of RAM, allowed modules to be coded efficiently and easily. With deployment of the collar, this processor is no longer suitable. A low-power variant from the same manufacturer can be chosen to allow for power gains while reusing most of the software drivers. These low-power processors have their own
interface issues, mainly the lack of RAM. RAM requires substantial power to maintain its contents, and typically, the low-power variants have less than half the ram of the high-power variants to save on power. This lack of RAM could force the re-factoring of several modules to reduce RAM usage. For example, the filesystem details are stored in RAM indefinitely, but after initialization there is no need for it to remain in RAM unless the filesystem needs to be adjusted (a very low probability chance given the current architecture).

4.2.3 RTCC addition

With the current system, there are limitations to the gains acquired by sleeping the processor; this is due to the need for accurate time keeping. While the mountain lions have behaviors that would enable the collar to be completely slept (i.e. sleeping and grooming), this can not be done in the current generation as the hardware timers do not have the ability to run in deep sleep. The use of the onboard RTCC (Real time clock and calender), necessitating the need for a low-power oscillator, would allow for time keeping in deep sleep. Limitations in the accuracy of this timer (most importantly, the inability to set an alarm shorter than half a second), will require significant re-factoring of the sampling code. Implemented properly, this addition could give long stretches of time where current draw is in the nano Amps range.

4.2.4 GPS Clock Tuning

Accurate timing is paramount to enabling behavior detection but onboard oscillators are limited in their accuracy. Microprocessor manufacturers are well aware of this flaw, and allow for tuning of the onboard oscillators. Without a known timing source, this ability is of little use. Luckily, there is accurate timing in the form of GPS which requires highly accurate timing to acquire a fix. It should be possible, either through a pulse per second output or the serial stream itself, to use the GPS to calibrate the internal clocks. This could either be done
before deployment or within the first couple days.

4.2.5 Onboard Flash memory

microSD cards are an incredibly versatile format for the collar given their recovery options. At the same time, the protocol on which they run require a significant handshake for each transaction. Without significant amounts of available microprocessor RAM to store sectors, streamlining this process is difficult. One possible solution is to put embedded flash memory on the PCB itself and use this storage for general data collection. When either enough time has passed or enough data has queued, the microSD card can be initialized and the data transferred to it. This transfer could possibly be streamlined through the use of a peripheral to peripheral transfer using DMA on the processor (available in both high power and low power variants).

4.2.6 Behavior Identification

While the random forest approach has yielded an algorithm that works well, it remains cumbersome and difficult to train. Alternative methods from the signal processing domain should be explored and possible alternatives.

For instance, a simple FFT of the accelerometer data could quite possible reveal markers into the animal behavior, and would not require extensive training sets. Wavelet analysis allows for better fine tuning of the identification, and present a rich area for exploration.

Lastly, within the area of non-linear classifiers, neural networks have performed admirably in the past. This should also be explored, though it suffers from many of the same training/validation problems that the random forest classifiers do.
Bibliography


