UNIVERSITY OF CALIFORNIA, SANTA CRUZ

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

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

Solar-Powered Smart Wireless Camera Network for Outdoor Monitoring

by

Kevin Mathys Abas

June, 2015

The thesis of Kevin Abas is approved:

Professor Katia Obraczka, Chair

Professor J.J. Garcia-Luna-Aceves

Professor Anujan Varma

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Solar-Harvesting Wireless Smart Camera Research System For Efficient Sustainable Outdoor Video Analysis

by

Kevin Mathys Abas

With the advancement of the Internet of Things movement, ubiquitous computing is now an accelerating field for embedded system researchers to pursue. Almost all of these new connected systems are equipped with sensors to gather useful information about their surroundings, and of these sensors the visual sensor presents the richest available datasets in comparison to its scalar counterparts. Deploying wireless sensor networks in remote environments makes it difficult to acquire power to these systems and many choose a battery-supplied source. After optimizing the sensor node for long battery life, many sacrifices are made in performance to delay the need for battery replacement maintenance.

Now, however, there is promising furtherance in energy harvesting and rechargeable battery capable sensor nodes, and the need for node maintenance may be obsolete in the coming years. Sensor network devices can now focus less on optimizing for battery life and instead increase the amount of on-board local processing making the device "smart". What defines a smart camera is its ability to process video footage locally instead of a central processing unit online or at a remote network sink node. Progress in the field of computer vision has brought the ability to identify complex objects and behaviors of monitored environments.

In this thesis, we present SlugCam, a solar-powered wireless smart camera network platform that can be used in a variety of outdoor applications including surveillance of public spaces, habitat and environmental monitoring, wildfire prevention and detection, to name a few. The system is built with off-the-shelf components which not only keeps it modular and low cost, but also facilitates its prototyping, rapid duplication, and evolution. SlugCam's on-board processing capability allows computer vision software to run locally which contributes to the system's autonomous

operation capabilities. Energy efficiency in SlugCam is accomplished both in: (1) hardware by micro-managing low-power components; as well as in (2) software by having the system's operation duty cycles automatically adapt to the current state of the battery in order to balance the trade-off between application-level requirements and power awareness.

Acknowledgements

I would like to first acknowledge my family for there support from the very beginning of my academic career to this moment in my life. To my mother, Vilma, for her persistence to tell me to continue working and caring motivation. Also to my father, Roger, for being not only inspiring mentor, but a supportive friend. To my sister, Kelly, for her support as well as her inspirational success at UCSC in her pursuit of a economics degree.

I would also like to thank my advisor professor Katia Obraczka for her support and feedback on all the work I have accomplished doing my research. I would like to express my gratitude for the hours she spent helping me make important career decision and how best to approach the plans I had for the project described in this thesis.

I would like to thank professor Anujan Varma and professor J.J. Garcia-Lunea-Aceves, the other reading committee members of my thesis, for their valuable input and advise on revising my final thesis submission.

Finally I would like to thank my colleagues in the i-NRG lab for their feedback and support. I made many life long friends in our lab, and together we had some experiences I will never forget. I will miss the collaborative community we had and will treasure what we accomplished together.

1 Introduction

Wireless Visual Sensor Networks (WVSNs) have been designed to meet strict hardware requirements for wireless sensor network deployments in indoor and outdoor scenarios[1]. In particular, cameras for outdoor deployment have been designed with power efficiency as one of their main design requirements. They also address the need for intelligently distributing available bandwidth to all the nodes in the network, especially in the case of dense deployments[40].Besides utilizing efficient hardware, WVSNs must also follow certain design criteria to satisfy performance requirements imposed by running applications while maximizing the systems lifetime by being energy efficient[43]. In particular, surveillance of public spaces requires not only efficient resource utilization to maximize service lifetime and availability, but also needs to ensure data being collected is handled in a secure way due to privacy concerns.

Visual sensor data has far higher storage and bandwidth requirements when compared to data produced by "scalar" sensors. Having cameras "stay on" and record live footage in outdoor surveillance scenarios is prohibitively expensive and may raise serious privacy concerns. Additionally, it is critical that only relevant data gets stored and/or transmitted to the end user to avoid consuming unnecessary network resources and overwhelming the monitoring user with large amounts of information, most of which is likely to be irrelevant. Efforts to achieve live streaming for multiple wireless smart cameras were successfully shown to be optimized for performance in [31], however the bandwidth limit and max node count still exist. One must assume operators either have to be monitoring all streams or having to filter through these recorded streams manually at some later point in time.

Smart wireless visual sensor networks, also known as smart camera networks, have advanced tremendously over the years and continue to be an exciting research topic. Smart camera networks are networks of visual sensors which take advantage of computer vision techniques as a way to not only filter relevant data to be stored and/or

presented to the user but also to save energy and communication bandwidth. Rich data provided by visual sensors, coupled with advances in computer vision techniques and ever increasing availability of computing resources continue to push the boundaries of the types of services and applications that WVSNs can support, including: environmental, wildlife, and habitat monitoring; independent living targeting the elderly, as well as surveillance of outdoor public spaces [44]. Local processing at the sensor nodes can have a huge impact on the amount of data that actually needs to be transmitted and archived. Whether in event detection or object identification, preventing false alarms by checking the data locally can be very helpful in a system's overall efficiency. Smart visual sensors can not only identify objects in a surveyed space, but also determine object specific behaviors, as will be described in Section 2. Computer vision and image processing research has been progressing at very fast pace; and, in order to be able to analyze visual data locally and make adequate decisions, wireless camera surveillance systems have been making use of increasingly more sophisticated vision techniques and algorithms. This has been possible due to the availability of sensing nodes equipped with higher quality cameras, processing, and storage capabilities.

The increasing availability and decreasing costs of solar-powered solutions have also been transforming the wireless sensor networking landscape. It has been enabling a wider range of applications in remote areas where access to the power grid is either non existent or too costly [49]. Solar-powered systems have also become an attractive alternative to traditional battery operated systems which typically require frequent maintenance and battery replacement. Rechargeable batteries, though currently don't keep charge forever, are being rapidly improved as well. A study showed a 5 times improvement in battery life in a new Lithium ion battery design[21], and the discovery of why the batteries degrade along with preventative opportunities were detailed in a publication just a year ago[12]. However, while solar systems take advantage of ubiquitous renewable energy, they also bring up other new challenges [11]. Notably, they must address challenges such variable sun exposure due to predictable

and unpredictable weather conditions. A recent publication [45] though showed a drastic improvement in solar efficiency using a new form of silicon surface, and such studies shed light on the ever growing potential of energy harvesting technology. It is important to note the other major challenge of wireless communication to these remote locations, which can be overcome with the help of long range antenna and satellite communication methods.

1.1 Motivation

In the middle of February of 2013 there were a series of crime incidents at UCSC that were troubling to the campus community so much so that the school's Chancellor was prompted to write an email expressing his concerns. Also, bike thefts and car thefts are frequent in remote walking paths and parking lots where it is impossible to deploy traditional monitoring networks. In an effort to combat the problem of safety assurance, the project to develop a more efficient capable wireless monitoring platform was launched. Since then the SlugCam smart camera research platform's applications have evolved and environmental monitoring and animal tracking deployments are also promising. The campus Information Technology Services (ITS) staff has given us locations where the system could be of use, and those trials are later described.

1.2 SlugCam Design Overview

To meet time and cost constraints SlugCam has been designed with readily available off-the-shelf devices to reduce cost and allow for rapid development. Each device has been chosen carefully to meet our power consumption efficiency requirements, and is later detailed in Section5. Unlike existing smart cameras that run solely on battery as their energy source, we have designed SlugCam to run efficiently on solar power while leveraging a rechargeable battery when sunlight isn't available. Another important design consideration was to ensure sufficient on-board processing capabilities which allows SlugCam to perform visual processing tasks locally, and thus be more selective

of the video it records and transmits, which in turn contributes to power efficiency. SlugCam's on-board camera is normally off and gets turned on by a passive infrared (PIR) sensor when motion is detected. Additionally, SlugCam is able to adapt its operation to the available energy remaining in its rechargeable battery, shown in Section 7. For the deployment and needs of our ITS sponsors, we also created management tool to remotely manage wireless nodes and view recorded video data from a SlugCam node as well. The ultimate goal of the system was to have it been deployed, be cost effective, and energy efficient so maintenance to the system would be minimal. The SlugCam Platform is a preferable choice to computer vision researchers needing to gather real video data in remote outdoor environments

1.3 Contributions

In this this, we present SlugCam, a wireless, solar-powered smart visual sensor network platform that can be used in a variety of outdoor applications. The thesis main contributions can be summarized as follows:

- To build upon Wireless Solar Smart Camera platform ideas and designs resulting in a cost effective, small form factor node.
- SlugCam uses an "open system" approach and is built with off-the-shelf hardware components which not only keeps it modular and low cost, but also facilitates its prototyping, rapid duplication, and evolution.
- SlugCam offers a unique management and user interface tool. Among other features, it maps and tags events to the corresponding video footage captured by camera nodes in order to present collections of fragmented information in a cohesive manner. Using computer vision techniques for object detection it currently allows a user to specify object behaviors for SlugCam to monitor and alarm if detected.
- SlugCam's energy efficiency is achieved both by fine-grained management of low-power hardware components, as well as by having the system's operation duty cycles automatically adapt to the current state of the battery in order

to balance the trade-off between application-level requirements and power efficiency.

- Unlike similar smart camera platforms relying on simulation results, a SlugCam node operational evaluation is completed and results are presented.

We show SlugCam's energy efficiency through a comprehensive power characterization. We also conduct a real-world deployment in a residential area on the UCSC campus which showcases that although SlugCam has additional processing capabilities when compared to its other wireless smart camera counterparts [1], the system could still run off solar harvesting sources and a rechargeable battery.

1.4 Thesis Organization

The rest of the thesis is organized as the following: Related systems and design practices are described in section 2. Then SlugCam's abstract features and smart energy efficient smart camera methodologies are explained in Section 3. Documentation on firmware, node software functionality and optional mesh networking capabilities is shown in Section 4. Section 5 is a complete description of the various hardware components of the SlugCam platform, while Section 6 briefly explains the Web app designed for deployment. In Section 7 we evaluate SlugCam, and we conclude in Section 8 along with previewing some possible future directions for the project.

2 Related Work

Over the past few years, smart camera systems have received considerable attention from academia and industry as reported in our recent survey [1]. In parallel, the field of computer vision has advanced rapidly creating new techniques in image processing. We mention behavioral identification publications to provide context for some of SlugCam's proposed applications and useful features provided by its user interface. We draw comparisons in this section from visual sensor platforms with solar and other similar capabilities as the SlugCam node as well to give some background to our design decisions.

2.1 Wireless Smart Cameras

Although our survey of wireless smart camera platforms for monitoring public spaces provides highly detailed comparison of recent platforms, we will list a brief summary of a few of them here to highlight popular trends:

- One of the best bandwidth efficient smart camera platform is *CITRIC*[7] and it achieves its success through the use of "sensor fusion" techniques. The system uses its low power on-board audio sensor to determine if and when the high power video sensor needs to be powered. Unlike SlugCam the system is battery powered, runs μClinux, and the authors developed a proprietary custom object detection scheme for the platform.

- The wireless camera platform in [17] reaches maximum power efficiency, allowing the node to stay powered for 3 months on a small 2200mAh 3V battery. The authors accomplish this with controlled idle and active management states and different levels of transmission video sizes depending on the level of activity and importance of the data.

- The *Wi-Flip* smart camera platform presents an efficient new design for an image sensor and with this they design a new custom processing algorithms that effectively detects environmental fire smoke [13]. The authors admit that the low resolution data and extremely low bandwidth capable radios are main issues.

- Also in some instances, designing an efficient network protocol (e.g., at the MAC layer of the networking stack) has shown to provide energy usage benefits like for *MeshEye* [22] and *OmniEye* [31]. It is important to also note how these systems are being powered and that none of them make use of energy harvesting or renewables (e.g., solar power).

- Using multiple sensors on separate nodes can provide additional information of the event (or lack thereof) to prevent false positives. The Human Situation Monitoring System (*HuSIMS*) does this on a very large scale by using numer-

ous sensors is dense city deployments [6]. Video sensors will detect abnormal movement during building fires and smoke sensors will be able to confirm that a fire is indeed the problem.

- Taking a more hardware focused approach to smart camera research, *Flexi-WVSN* also does an extensive evaluation of previous wireless visual sensor networks [46]. Realizing it was hard to stay up to date with hardware advances, *Flexi-WVSN* has developed a modular system so no sub-component is dependent on any other piece of hardware. Rather than a two microprocessor system as SlugCam has, they chose a two radio system to achieve energy efficient gains. When smaller data files need to be sent, the system can switch to a more energy efficient Zigbee radio, otherwise they use WiFi as a more reliable communication protocol to transmit video data for faster transmission.

- In surveillance situations where wireless communication is intermittent, delay-tolerant networking (DTN) techniques are proposed as a solution by authors in [42]. Even if communication is readily available, many visual monitoring applications don't require 24/7 high resolution video footage.

Of the smart camera examined in our survey our CPU processor speed exceeded those surveyed, mainly because of the trends to extend battery life. Now however, visual sensor platforms will need to run more computationally resource heavy software to adopt newly developed computer vision schemes.

2.2 Behavior tracking Vision Techniques

Computer vision techniques have advanced considerably and are able to not only identify objects accurately but also detect object behavior. SlugCam was designed with the intent to run more power visual analysis algorithms, and systems with higher local processing power will be able to execute the items described in the survey of tracking human behaviors with computer vision [23]. Researchers have also been using symbolic information from object trajectory behavior in the video footage like SlugCam in order to extract more meaningful information from the environments

they monitor [50, 2]. The *CamInsens* system demonstrates behavior detection in a distributed outdoor network, but without power efficiency stated as a primary goal.

2.3 Solar-Powered Wireless Visual Sensor Networking

Many non-visual sensor networks have been deployed outdoors with the ability to use solar as a reliable energy resource. Solar power combined with rechargeable batteries allows sensor networks to be less reliable on the power grid [49]. UC Berkley has developed the world's largest outdoor solar sensor network testbed with 557 nodes named *Trio* [11]. *FireWxNet* [20] is a wireless visual sensor network with solar capabilities which adapts to available energy resources and battery charge levels by adapting its duty cycling. While their visual sensor nodes do not perform visual processing, the authors claim that the system's auxiliary sensor nodes provide enough information to allow the user to then decide if they wish to receive a low resolution live video stream of the monitored area. In contrast to SlugCams use of 802.11 WiFi to transmit high resolution video data, FireWxNet uses 900Mhz radios to achieve long range communication. Doing this however sacrifices the ability to perform larger data transfers. A list of recently developed solar-powered smart cameras are listed with a discussion of similarities and distinctions:

- The system with the most similarity is described in both [35][36] and has both solar capability and an analog Passive Infrared Sensor to assist the systems custom computer vision software. It is important to note that their platform does not properly test video transmissions with a wireless protocol or solar power capabilities for deployment, but instead shows simulation results with many assumptions about power consumption behavior using solar harvesters that ignore real world occurrences. SlugCam has the follow important design distinctions:

    a) A Faster processor for vision analysis and a developer friendly operating system for rapid evolution of the node software.

b) An on-board current sensor for real-time power awareness.

c) A higher capacity rechargeable battery to account for extended periods of low light.

d) Low-powered WiFi module for higher bandwidth, with tested mesh networking capabilities, described in Section 4.

e) Larger solar panel and smart battery charger for battery charging and system energy consumption simultaneously.

- Although the WVSNs in [32] doesn't use computer vision, and interesting technique of clustering solar-powered WVSNs to improve battery life performance is proposed and evaluated using measured seasonal conditions.

- Efficiently distributing video capturing detailed in [8] and with the use of MIMO communication techniques grouped cameras are shown to have energy savings of 30-50%.

- *SensorCam* [9] does have on-board current monitoring circuitry, but the entire system in included on a custom printed circuit board making it difficult to change hardware components.

- The smart camera platform using a solar power source in [3] focuses on designing the system with the Zigbee communication standard[51], and the authors also mention the bandwidth limitations with range limitations.

SlugCam not only tries to achieve efficiency goals using different hardware and software components, but different parts of the system are extremely modular to be able to rapidly adapt to new hardware releases such as switching to the more powerful Raspberry Pi model 2 or integrating a more efficient battery charger.

3 Methodology

3.1 Adaptive Hardware

One of SlugCams notable features is that it includes two processing units: an MSP430 and a Raspberry Pi. The low-power MSP430 micro-controller [27] manages

the amount of time that the more power-consuming Raspberry Pi stays on. This functionality, further described in Section 5 is accomplished by having the MSP430 monitor PIR motion sensing while the Pi controls the rest of the system. When no motion is detected, only the MSP430 is "on" and all other components of the system, including the Pi, are "off". As will be shown by our power characterization experiments, this interrupt triggered approach yields substantial power savings, which allows SlugCam to use smaller size solar panel and battery, and yet run more processing-intensive computer vision algorithms on-board.

Every external module used including the PIR, camera, and WiFi have the ability to be powered down as well, which allows SlugCam to adapt its operation according to the current state of its battery. For example, when not active, the WiFi module is in low-power sleep state until it receives a signal from the Raspberry Pi to wake up. SlugCams range of duty cycles define the sequence of tasks executed by each SlugCam component and their corresponding power states. As previously pointed out, SlugCam's main processing device is the Raspberry Pi, (or the "Pi") [16], which runs the nodes main functions including its visual processing tasks. Due to their high system resource usage, visual processing tasks are not always executed during SlugCams duty cycles. Depending on the current state of its battery, a SlugCam node will decide whether to perform visual processing on-board. For example, suppose that during the night there was considerable activity and, as a result, the system was frequently active, thus consuming more energy. The next day is cloudy/rainy so the solar power scavenging component could not charge the battery sufficiently. Consequently, SlugCam will not run its vision software, or choose to take snapshots of the scene, until its battery is charged beyond a certain level. This battery charge threshold is a configurable system parameter and influences the trade-off between energy efficiency and event detection requirements[43].

Protocol Name Data Rate	Frequency	Outdoor Range	Current Consumption	Max
Zigbee	2.4GHz	30-100m	Tx:25-35mA, Rx:20-30mA	250kbps
z-Wave	900MHz	$\leq$100m	Tx:30-40mA, Rx:20-30mA	40kbps
6LowPAN	2.4GHz	$\leq$200m	Tx:20-35mA, Rx:12-25mA	200kbps
Bluetooth BLE	2.4GHz	$\leq$50m	Tx:5-10mA, Rx:5-10mA	1Mbps
WiFi 802.11g	2.4GHz	$\geq$800m with antenna	Tx:$\geq$120mA, Rx:$\geq$40mA	54Mbps

Table 1: Well know statistics compared from popular wireless communication standards commonly used in embedded systems.

3.2 WiFi vs Other Embedded Wireless Protocols

While WiFi has been one of the most ubiquitous wireless communication technology, its energy efficiency has always been an obstacle for its use in embedded systems. The Zigbee wireless standard (802.15.4) and Z-Wave standard have been advertised as being both cost effective and extremely energy efficient[51, 4]. Smart Camera deployments using Zigbee have confirmed our theories on bandwidth limitations [3, 34] even though they favored the protocol for its low energy consumption. Bluetooth BLE has surpassed others in energy consumption savings with some modules achieving as low as 5mA for both transmission and receiving [41]. Both 6LowPAN [18] and Bluetooth [5] were ignored for both range coverage and protocol ease of use for both infrastructure and ad-hoc needed requirements. Recently, however, energy efficient WiFi modules have been developed specifically for embedded applications. SlugCam uses the RN-171 WiFi module which can be put into a low-power sleep state when not in use consuming just 4μA [24]. We chose the RN-171 not only for its low power consumption and high bandwidth, but also because it has the capability of operating in ad-hoc mode This ad-hoc feature will allow for networks to be extended farther distances, further assisting the systems ability to operate in more remote locations. One of the shortcomings of low powered WiFi modules is the maximum bandwidth allowed by the UART serial communication line prevents us from making full use of WiFi's transmission speeds. We believe, however, that favoring energy efficiency over transmission speed is more beneficial in the case of SlugCam. In addition to the WiFi module efficient features, the raspberry Pi current consumption drops on average an additional 200 mA when the USB hub is not being used for a standard USB WiFi dongle.

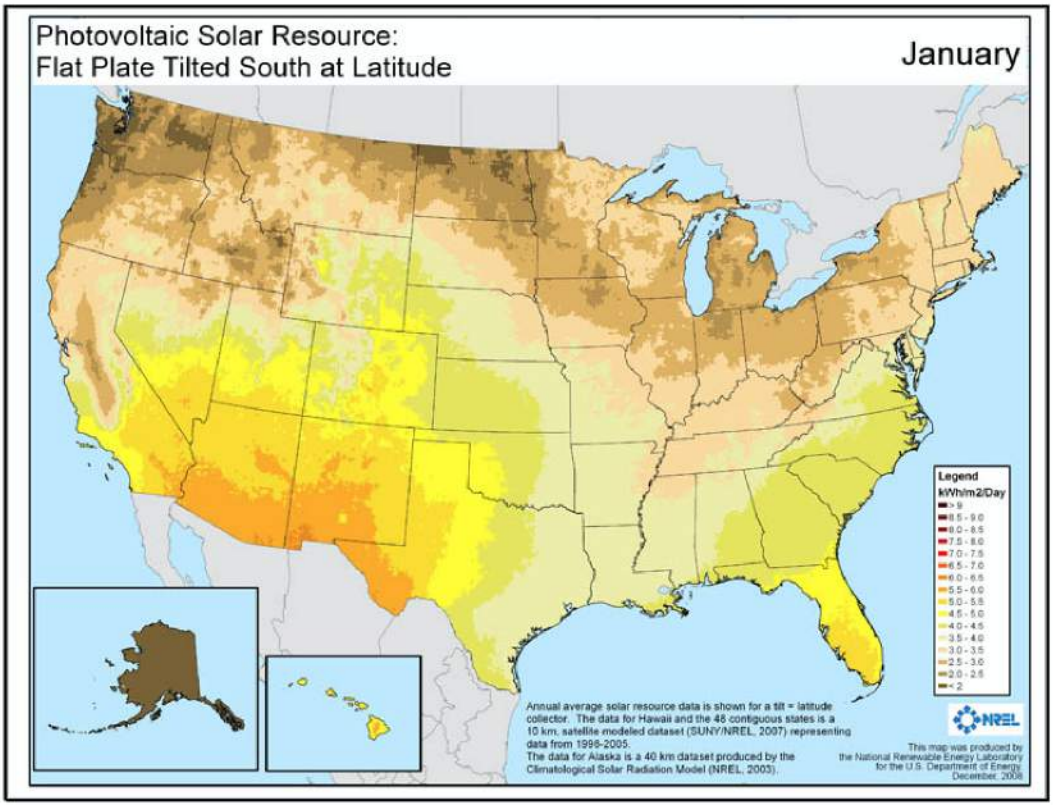

Figure 1: Average solar radiation data for the United States for January, which is the month having the least amount of sunlight in every year.[39]

3.3 Solar Harvesting Techniques

As previously mentioned, one of SlugCam's main goals is the ability to not rely on the power grid or wired communication infrastructure, and operate autonomously and unattended for extended periods of time. Equipped with solar-powered batteries and directional WiFi antennas, SlugCam is completely wireless and thus can be deployed in remote, hard to access areas where access to the power- or wired communication infrastructure is either not viable or too costly. Unlike a number of existing sensor network deployments [46, 1], SlugCam should very rarely need to be serviced for battery replacement as shown by our experiments. Pervasive wireless communication, lower-cost computing technology and solar power solutions make SlugCam increasingly viable and affordable [48] in countries having good available solar energy like the United States shown in Figure 1. Unlike systems that offload large amounts of video footage to a more powerful central server, on-board visual

processing capabilities contributes to SlugCams autonomy, lower energy- and communication bandwidth requirements. These are key features that enable deployment in outdoor, remote environments, and significantly increase the range of applications for SlugCam.

3.4 Open Source Hardware

One of our main goals in the SlugCam project is to provide a platform that other researchers and the community at large can use and extend. As such, our system is both flexible and easily reproducible. As previously pointed out, the current SlugCam node is equipped with two sensors, namely the camera and the PIR, but can be extended with other sensors in order to adapt to different applications and their requirements. As will become clear from our description of SlugCams software system, its design allows it to easily adapt to new data collection and processing methods in a way that does not require modifications to the core components of the system. We also designed our system in a modular fashion so that when modifications to the core components must be made, they are constrained to a module of the system and do not require knowledge of how the rest of the system works. This has been proven to be an effective strategy in designing smart cameras as they need to stay up-to-date with performance and technology upgrades [46].

Another one of our main design decisions was choosing to keep both hardware and software designs as open as possible. Not only do we make our code available on a public online repository, but we have also used hardware and software that are both open source and very friendly to modification. We ensure that every part of the software we have developed is free and open source, including the network software platform, the database software, the Web frameworks, and utility libraries. In the same spirit we have also used off-the-shelf components whenever possible and will release schematics for any custom hardware so that the system can be easily reproduced. This helps in increasing both the flexibility and reproducibility of our system as well as encouraging developers to extend it and apply it to other

applications.

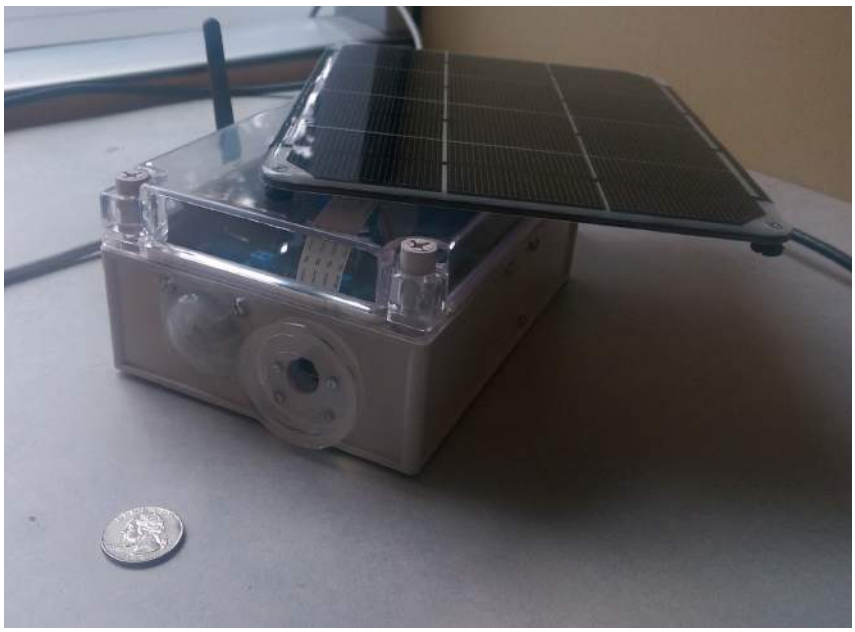

Figure 2: The second iteration of the SlugCam prototype in its weatherproof enclosure and re-positional panel on top.

4 SlugCam Platform Software

Previously on The SlugCam node we had installed our own custom embedded Linux distribution to optimize performance and processing speeds using the Buildroot tool [33]. Buildroot allows users to build a customized embedded Linux operating system to fit the needs of an application. From Linux packages and tools such as OpenCV[29], all the way to more complex hardware specific configurations required by the devices hardware architecture, Buildroot provides a complete customizable solution. Creating a SlugCam-specific Linux operating system with all the packages we need (and none that we do not need) really boosted the Raspberry Pis ability to process tasks quickly and efficiently.

However we later realized the need to cross compile any new libraries and the annoyance of missing core OS utilities made us decide to instead install the Arch Linux OS on the Pi [19]. Only Sacrificing miliseconds of boot time, we found Arch Linux

to be equally as powerful and capable as the custom Linux distro.

Another innovative feature of our system is its robustness to arbitrarily frequent and long-lived disruptions in communication between its different components. To this end, we built all communication protocols following an "opportunistic" approach. For example, the video and message servers constantly wait for incoming connections from camera nodes and expect that those connections could be dropped at any time. Any messages that need to be sent to the camera must wait until that camera comes online; and as soon as the camera does come online we begin to send all outgoing messages. This opportunistic communication framework further increases camera autonomy at the cost of immediate communication ability. It allows for many interesting optimizations, such as reducing communication in low power situations and when only non-interesting data has been collected. Cameras can also be set to come alive on a timer to receive important control data from the network if needed in order to manage the communication latency inherent to this type of design.

4.1 Raspberry Pi Management Software

Leveraging the Arch linux operating system we were able to easily configure system functionality to meet the application requirements and set up "built in" linux security measures such as user permission policies.

4.1.1 Inter-Process Communication

Rather than build a custom process manager, controlling all of the software modules, we used Arch Linux's built in Unit and Unit management feature. All units can by default be enabled on bootup, and each unit manages its connected external device. Cooperation and the need for processes to communicate a system global notification, or to adapt to a new duty cycle, is accomplished by using UNIX domain sockets. Without any performance loss, the processes use reliable stream communication protocols by reading and writing to a UNIX domain sockets using the similar syntax as a traditional UNIX network socket. Since the data is communicated through a

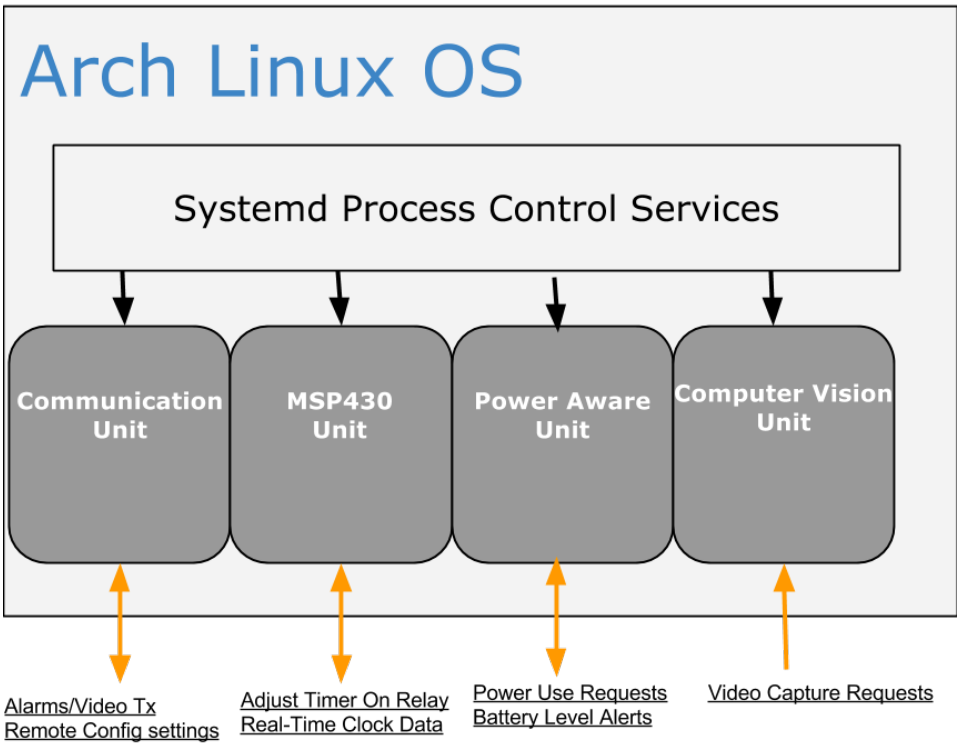

Figure 3: SlugCam's High-Level Hardware Architecture.

stream file (e.g. *"/tmp/paunix.str"*) file permission settings are easy to enforce, and when a process calls *sendmsg()/recvmsg()*[1] it grants the other process permissions to access the file descriptor.

4.1.2 Relay control Unit

Since the node is also responsible for balancing the trade-off between application requirements and power efficiency, it needs to manage its operation accordingly. This service is responsible for looking at battery levels and commands from other system Units (and possibly a remote user) other future requirements and conditions) to determine the appropriate duty cycles at given points in time. This is where we can facilitate many of the varying video capturing modes such as immediately sending video with action in preset areas, allowing video recording on a timer, and taking still shots instead of video when battery levels are low. Also because the Raspberry Pi does not have a hardware clock, the real-time clock is set and managed by the

[1]For a complete functional description of UNIX domain sockets please see [47]

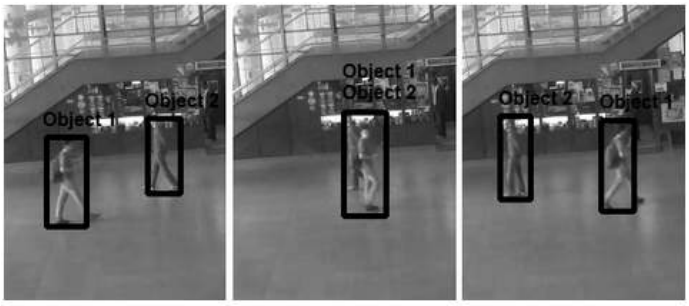

Figure 4: Demonstration of a Occlusion with SlugCam's background subtraction image processing being executed.

msp430.

4.1.3 Computer Vision Unit

The current prototype performs object detection using well-known background subtraction mechanisms due to their adequate cost-performance trade-off. SlugCams image processing module works as follows: when the system is initialized, the object detection module keeps taking pictures with 1280 x 720 resolution and re-sizes them to 426 x 240 so they can be processed. As described by Johnson and Tews [30], it uses the MoG method to extract the foreground at an average of 3 frames per second (fps). With the resulting foreground, we find the contours and then a bounded box is applied following the contours' limit(See Figure 4. To remove noise and false detection, a step is added to ignore objects of small size. Object classification is one of the most computationally complex tasks in computer vision systems; this is because it uses training techniques to classify the objects based on a previously stored database. In our case, to reduce complexity and add more flexibility, we classify the objects by considering their dimensions. This unit doesn't operate autonomously and only receives inbound requests, shown without the bi-directional request flow in Figure 3

4.1.4 WiFi Transmission Unit

This means that communication unit is implemented only once in our software package and each program does not have to worry about any network functionality and

how it changes with future development. Changes can be made to protocols and wireless card interface without needing to modify multiple processes. For example, if we later decided to adopt a different wireless module, we could rewrite the unit and, as long as the API remained consistent, it would be an easy drop in replacement. Another advantage is that acts as the gatekeeper to the wireless module. This is important as we are communicating over a serial connection, which means that would be difficult to coordinate multiple processes attempting to use the serial interface at the same time.

4.1.5 Energy Consumption Unit

The power monitoring unit not only responds to requests of system battery levels, but if the battery reaches certain levels it will send the appropriate requests to other processes to alter duty cycle behavior or limit power hungry activities. Battery levels are approximated using the battery state signals given by the solar charger module and periodic monitoring of node energy consumption using the on-board current sensor explained in Section 5

4.2 MSP430 Firmware

The microprocessors code can be viewed in Appendix B. The MPS430 Firmware includes the ability to control the Raspberry Pi's power state, run a real-time clock, and receiving lengths of time over a serial SPI communication link. These time lengths, are period lengths before a system timer fires a interrupt service routine triggering the relay to cut power and put the system in a low powered sleep state.

4.3 Mesh Networking Capabilities

In a recently submitted paper [38] we showcased our implementation of the DSR[2] routing algorithm to route SlugCam traffic in an ad-hoc network. Titled SCmesh, the network software implementation gives the SlugCam system the ability to deploy in ad-hoc deployments, helping communication challenges presented with deploying

[2]DSR stands for Dynamic Source Routing and is a popular on-demand ad-hoc routing scheme.

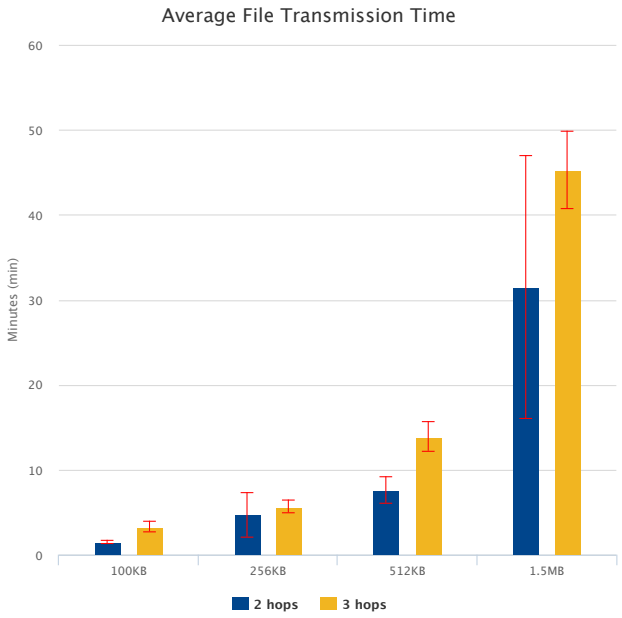

Figure 5: Average transmission times of different video file sizes with standard deviation shown in red.[38]

Infrastructure enabled WiFi nodes in remote outdoor locations. Transmission tests were completed of different captured video files, showcasing multi-hop, multi-path, and cost oriented routing features. Promising results showing transmission delay of captured video files can be seen in Figure 5. In addition Multipath packet delivery behavior using our power aware DSR approach with battery level generated costs is illustrated by Figure 6

5 SlugCam Platform Hardware

At the core of SlugCam is the Raspberry Pi (or sometimes referred to as the Pi), an open-hardware device intended to be used by the educational- and research community [1]. The high level block diagram of the system is illustrated by Figure 7

5.1 MSP430 Relay Circuit

The MSP430's ability to run on .05mA on standby and nearly 5 times less current when sleeping makes the MSP430 quite attractive in the context of systems where

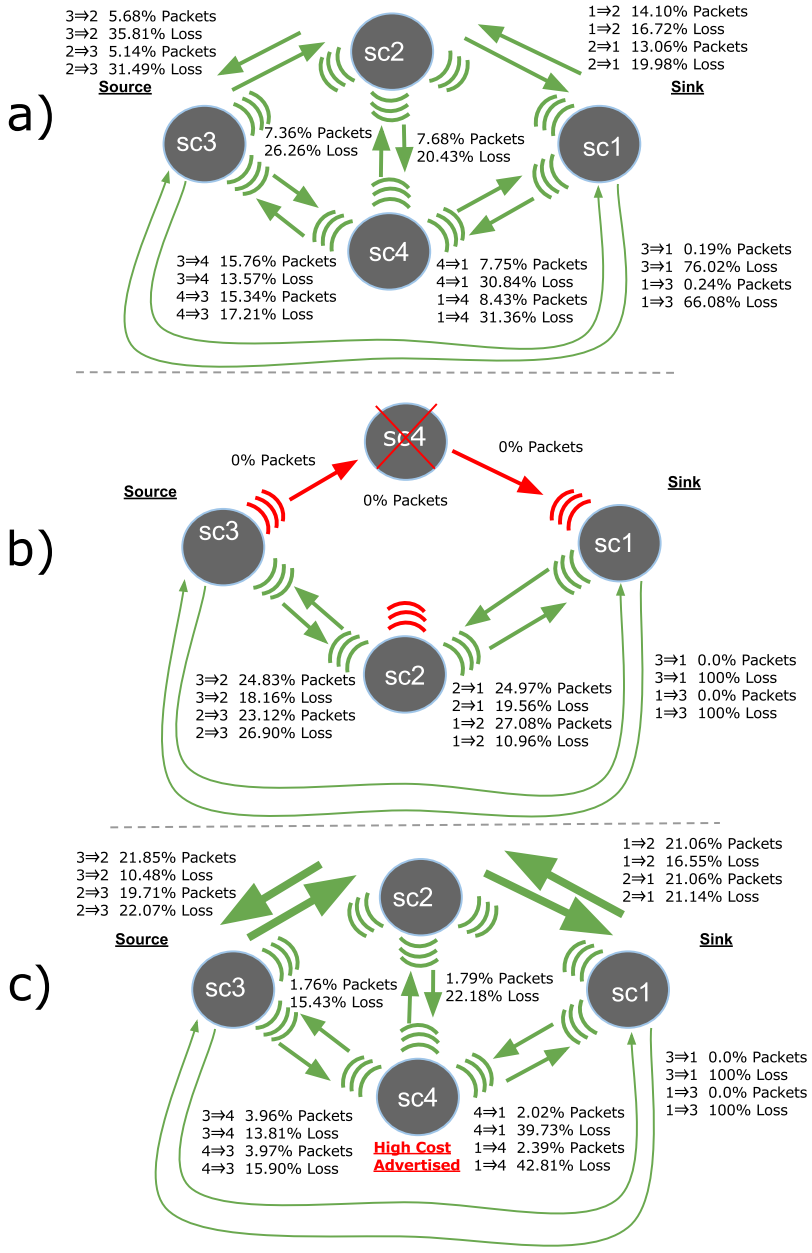

Figure 6: a) The multipath packet flow distribution and Loss ratio of 4 SlugCam nodes all making advertisements of equal cost. b) A camera node becomes inactive due to battery depletion and the packet flow with packet loss is shown. c) SlugCam 4 node advertises a higher cost due to low battery levels so more traffic is routed through node 2.[38]

power-efficiency is of critical importance. In the case of SlugCam, by controlling how long the Raspberry Pi and its peripherals stay on, the MSP430 is able to decrease the overall power consumed by SlugCam's nodes substantially. The MSP430 accomplishes this using an on-board timer and an external mechanical relay acting

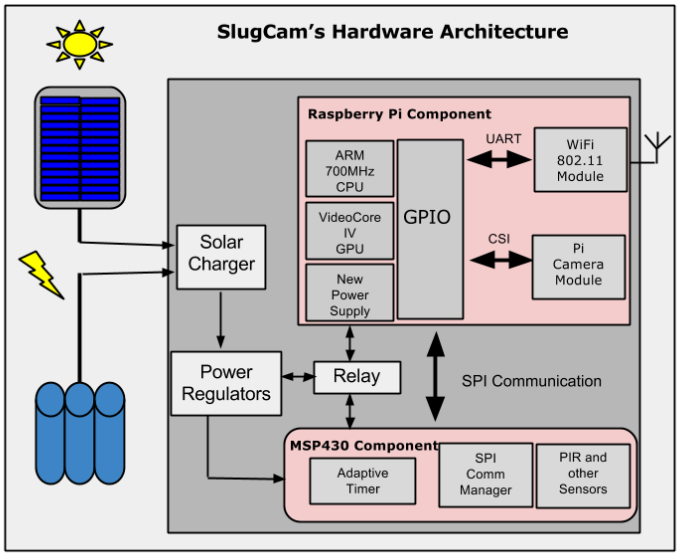

Figure 7: SlugCam's High-Level Hardware Architecture.

as a switch. The Raspberry Pi can dynamically change how much time it stays on by communicating to the MSP430 over a Serial Peripheral Interface (SPI) connection. To prevent time drift , a 32Khz crystal clock is added which has less error in large temperature changes in outdoor environments. The relay then waits for a signal that the "Pi" has successfully shut down before cutting the power from the system. This reduces the power usage greatly as long as the MSP430 does not wake up the RaspberryPi too often due to false positive movement. The MSP430 may receive messages from the RaspberryPi over the serial connection interface, telling it to relax its constraints on detecting movement. For example if a wild animal is causing the alarm, the MSP430 can start a timer that can vary in length for when it should again be sensitive to movement.

5.2 Camera Module

We developed our system using the newly released RaspberryPi camera module(photographed in Figure 8, which has an available API in C language and is capable of capturing 1920x1080 resolution color images with 30 fps rate as a maximum. The Pis image sensor will also make use of the Pis on board CSI Camera port and will allow us to not have to make use of a USB Webcam. We have been thus able to sig-

Figure 8: The newly released Raspberry Pi compatible camera module.

nificantly reduce the RaspberryPi's power consumption by not drawing any extra current from the USB hub. The Pi then consumes less power overall by relying on the CSI interface.

Last year the Raspberry Pi Foundation also released an advanced camera module that communicates to the Pi using the on-board Camera Serial Interface (CSI). The camera has the ability to take high resolution (2592 x 1944 pixels) images and capture 1080-pixel video at 30 frames per second [15].

5.3 WiFly Module

Due to bandwidth and communication range requirements as well as the quality of the image data we wish to record, we decided to use WiFi as our wireless protocol. Zigbee (or IEEE 802.15.4) may have been a better choice in terms of cost, but its power efficiency gains were not as convincing. Manufacturers of WiFi embedded modules are making large strides in reducing power consumption and this is very important for our energy constrained system. For example, we are currently using the RN-174 WiFly module from Microchip which will be able to go into sleep state and consume only 4μA .

5.4 Panel Choice

Unlike traditional battery powered smart cameras, SlugCam will rarely require battery replacement as it uses solar harvesting techniques and rechargeable batteries.

In order to size the solar panel and battery for our power requirements while keeping cost and form factor low, we researched information on solar radiation levels in the USA and made accurate estimations of SlugCam's power requirements. More specifically, for our planned SlugCam deployment, we looked up solar radiation in Santa Cruz, CA. This data was found on the National Renewable Energy Laboratory (NREL) website [39]. The NREL Web page shows solar radiation for every month at peak, average, and minimum levels and at every possible orientation of PV cells relative to the sun. Since one of our main goals is for the system is to operate unattended for long periods, we conducted worst case analysis, i.e., we needed to make sure that even with extended periods of little sunlight, SlugCam could still operate. SlugCam even if in degraded mode, i.e. offering only minimal service, will still be effective for its needed applications. Knowing that the worst case sun exposure is in the winter months, we chose the month of January with a PV cell orientation facing south as the basis for our analysis which was the most effective for our location. The energy produced by a photo-voltaic cell is directly proportional to the active area of the cell, the amount of light falling in that area, and the conversion efficiency of the cell.

$$\text{EnergyOutput} = \text{SolarEnergy} \times \text{PanelareaPV} \times \text{Efficiency}$$

Current PV cell technology differs in efficiency, construction material, and cost. Today the most known PV cells materials are Crystalline silicon (e.g., mono-crystalline silicon and poly-crystalline silicon) and Thin Film (e.g., amorphous silicon, cadmium telluride, and copper indium gallium selenide) [10]. We found a mono-crystalline solar panel with 6V and 19% of efficiency. Using the data from our solar radiation research we found Santa Cruz in the month of January receives 2-3 kWh per day.This unit of measurement represents the amount of energy accumulated over 1 square meter in a 24 hour period. Calculating how much energy can be converted to energy with the panel chosen of size 220mm by 175mm gave us the total energy produced for a whole day in our area.

5.5 Battery Choice

$$\text{BatteryCapacity} = \text{Consumption(Amps)} \times \text{NightUsage(Hrs)}$$

Now using the above equation and considering the Raspberry Pi consumes around 300 to 400 mA and during 8 hours per day on average, we have estimated the Pi taking 3200mAh total per day. Converting our current consumption per hour to joules showed our system needs less than what would be provided by a solar panel with all day direct sunlight. Since PV cells can only absorb energy when there is sunlight and an efficient provided angle, batteries are needed to power the system when there is little to no sun. Lithium-ion (Li-ion) is currently the most promising battery system: it is used for portable consumer products as well as electric power-trains for electric vehicles. Its main drawbacks are that it is more expensive than other types of batteries (e.g., Nickel and lead acid systems) and needs protection circuits for safety. Another critical factor when choosing the battery was its capacity, i.e., how much energy it can store in ampere-hours (Ah). This is important because SlugCam will rely on its battery as its power source when there is no sun. Thus, our battery capacity estimation was based on how much time the target location for our deployment receives no solar radiation. Using on our capacity estimations and battery type comparison, we chose a 3.7V 17600mAh pack of Li-Ion batteries which will allow the system to last up to 4 days without sun and using the maximum performance previously specified.

5.6 Battery Charger

With the sub components we have chosen and our adaptive power aware system, SlugCam Will last longer than this for certain less intensive applications. In order to charge the battery adequately and safely, we chose to use Microchips MCP73871 charger [28] which allows us to both charge and make use of the extra energy available when full sunlight is available. The charger also had other useful features such as temperature monitoring for the battery which prevents charging the battery under extreme weather conditions. Status signals are given for current battery conditions like when the battery has finished charging or when the battery has a very low charge

level.

5.7 On-board Current Sensing

The Pi also records live current consumption data to be processed by SlugCam, allowing it to be a more power-aware system, and to adapt to what battery charge remains. Knowing its current usage and available battery level, the system can estimate how much time is available before it needs to enter a sleep state waiting for available solar power. The current sensor [37] also allows us to determine what generic power consuming state the system is in. The idling state is when both microprocessors are on and idling, but all auxiliary sensors and the WiFi module have been disabled or put to sleep. When we discuss SlugCam being in a high powered state, the system is using both the WiFi module and auxiliary sensors like the camera module actively. The low powered sleep state, refers to when only the PIR sensor is idling and the Pi is powered off, and the MSP430 has been put into its own sleep state waiting for a hardware interrupt from the PIR sensor to wake SlugCam up.

5.8 Daughter Card and Enclosure

SlugCam includes a custom PCB daughter-card to house its external devices and available I/O ports. The daughter-card plugs directly into the Raspberry Pi and improves robustness and reliability of the systems hardware; it also reduces SlugCam's form factor. Future revisions of SlugCams daughter-card may include additional sensors to address the needs of a wider range of outdoor monitoring applications. In Figure 2 the weather proof enclosure is shown as well, with the panel detached to allow for optimal positioning.

6 User Interface

SlugCam also includes a Web-based tool for managing its network of wireless camera nodes and an innovative user interface for managing captured data. We consider this to be an essential part of smart camera systems, and yet something that is

often overlooked by smart camera system designers. For instance, smart camera networks such as SlugCam have the potential to collect large amounts of data that is fragmented over time and space. This presents unique challenges in the design of an effective, yet easy-to-use interface. SlugCams management tool allows end users to configure nodes, send them control signals, and access collected data. Its graphical interface makes the tool easy to use and very effective as a way to access and visualize collected data.

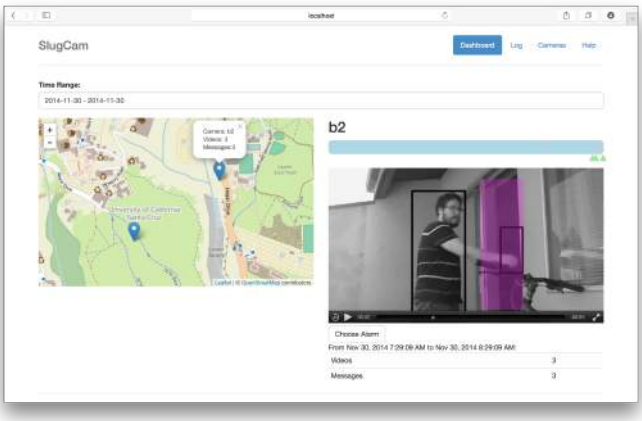

Figure 9: Screenshot of the dashboard of the client application featuring the activity bar, map, and alarm area selection.

6.1 Node Management

Since SlugCam deployments may consist of potentially large numbers of nodes, the system's management tool displays all currently deployed nodes in a map of the area being monitored. This view is called "dashboard" as shown in Figure 9. We also built what we call the activity bar which displays system activity over a period of time. This is an important aspect of our visual presentation as it allows the user to correlate collected tags to the videos in which activity is happening in an intuitive way, but without hiding any information.

6.2 Setting Monitoring Alarms

Another important feature is the ability to set certain areas within the camera's field of view that will trigger alarms. When an alarm is triggered it could mean that the

user is notified and the recorded video is instantly sent to the server, though we plan to allow users to define actions taken when alarms are triggered. The interface we have developed is intuitive and allows the user to select an area over the recorded video taking into account the current position of the camera. The alarm selection interface is currently available anywhere that video is displayed, and is demonstrated in our screenshot of the dashboard in Figure 9.

7 Deployment Tests and Experiments

During the design phase of the project we took some preliminary current consumption readings using a multi-meter and isolating parts(see Figure 2). In order to prove the systems has enough energy efficiency to deploy the system in real world scenarios though we needed to do some accurate power analysis of the system using the on-board hall-effect sensor. Using a simple 10 point moving average filter, we were able to smooth out our current consumption data for readability without sacrificing accuracy. We provide a proper current consumption analysis of the system both evaluating the duty cycles we defined and analyzing how the system achieves efficiency through some example application scenarios.

System Components	Current Usage
Raspberry Pi Idle	322.7mA
Pi with USB WiFi Idle	431mA
Pi With USB WiFi Tx/Rx	482mA
Pi with CSI Camera	532.7mA
MSP430 idle	230μA
MSP430 SleepMode	1μA
PIR sensor	1.63mA

Table 2: Measured current usage of some of SlugCams hardware components.

7.1 Energy Gains using the MSP430

One well-known and widely-used sensor network power savings techniques is to keep the system in a low-power state whenever it is idle, i.e., not executing any specific task. In SlugCam, we go a step further and use a low-power micro-controller (the MSP430) to control: (1) a PIR motion sensor to wake-up the Pi when it detects motion and (2) a relay circuit to cut power to the Pi based on an on-board MSP430

timer, which can be dynamically set by the Pi. In order to show the power savings achieved through our design, we run SlugCam for 5 minutes during which the system wakes up 6 times. When it wakes up (triggered by motion detected by the PIR), the Pi turns on the camera for 30 seconds before going back to sleep. As baseline, we run the system uninterruptedly for 5 minutes. Figure 10 shows the current consumed in the two runs. We use a 10-point moving average filter to smooth out the sampled current consumption data for readability without sacrificing accuracy. Calculating the area under both curves, we observe that the system saves around 30% when monitoring a relatively busy scene. The dips in the curve show that in the low power state, the system is consuming between 60 and 70 mA, while in the full on state, it consumes around 400mA. We note that the power savings can be even greater if we take advantage of the MSP430 deep sleep mode when it only requires current in the order of μAmps. We also performed a comparison between the recent Raspberry Pi B+, which SlugCam is using, and its precursor, the Raspberry Pi B. We found that latter consumes on average 150mA with no connected components and the new B+ model consumed around 95mA.

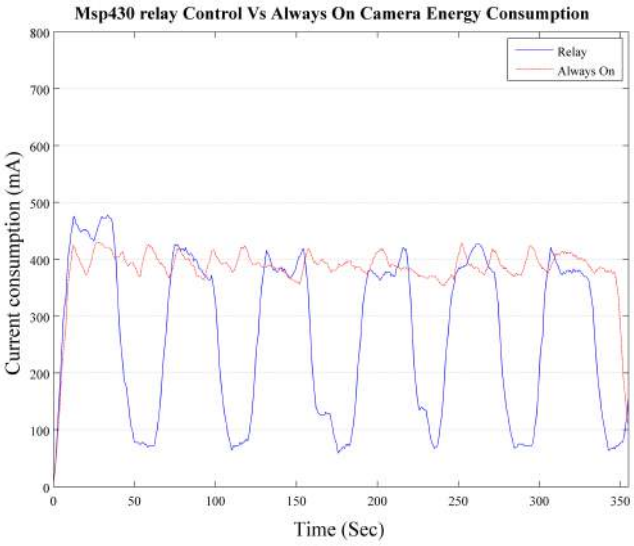

Figure 10: SlugCam power consumption showing power savings by using the MSP430 to control the PIR motion sensor and power relay.

7.2 Duty Cycle Examples

Duty cycling has been adopted by wireless sensor networks in general, and visual sensor networks in particular, as an effective energy savings technique [46] [1]. It typically works by having sensor network nodes switch to low-power states as much as possible in order to save overall energy consumed. SlugCam uses what we call adaptive duty cycling which allows the system to use current battery level information to guide its choice of duty cycles. In particular, when battery levels are too low (i.e., lower than a given threshold), the system avoids duty cycles that power hungry. Clearly, this typically comes at the price of degrading performance. This trade-off between performance and energy efficiency can be adjusted to meet application requirements by controlling the battery level threshold parameter.

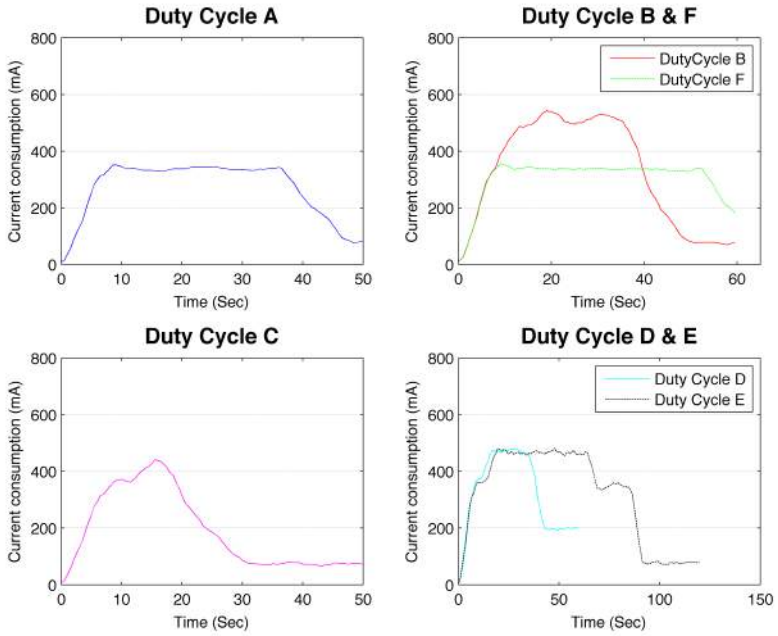

Figure 11: Power consumption characterization for SlugCam duty cycles.

We show SlugCams power consumption characteristics for the six different duty cycles described below:

Duty Cycle A: This duty cycle illustrates the case of the remaining battery level being below the threshold. In this particular experiment, the system wakes up and runs system services in which it determines the battery remaining is below the

threshold.

Duty Cycle B: In duty cycle B, the Pi is turned on by the PIR sensor and triggers the camera. The camera is turned on but does not run any computer vision functions. The WiFi module is also asleep as this duty cycle only records video and stores it locally. An alternate, lower power duty cycle to this one could take a high resolution image, instead of recording a video.

Duty Cycle C: Duty cycle C is the same as duty cycle B except that the system uses computer vision algorithms when recording data. In this particular instance, noticing the data is not important (based on the computer vision results), the system quickly transitions to low power state and the MSP430 cuts power to the Pi via the relay. As shown in the bottom left graph of Figure 11, at low-power state, the system (essentially the MSP430 in idle mode) consumes around 70mA. However, as previously pointed out, power consumption can be substantially decreased by having the MSP430 go to sleep.

Duty Cycle D: Duty Cycle D is very similar to C with the only difference being that in C nothing is recorded while in D the video is stored locally. This explains the difference, albeit small, in power consumption between the C current consumption curve (in the bottom left graph) and Ds (in the bottom left and right graphs of Figure 11, respectively).

Duty Cycle E: The most complex and longest running duty cycle is E, during which SlugCam records video (longer than D) with computer vision turned on and then immediately transmits the data. In Figure 11, for the E current consumption curve in the bottom right graph, the first drop in current consumption happens when the camera turns off and the WiFi module is on in order to transmit the video file to our Web server. The second in current consumption drop corresponds to the system going back to its sleep state with the MSP430 in idle state.

Duty Cycle F: In duty Cycle F, the system wakes up on its own to ping the Web server for any requests or to report status updates. This time is also used to transmit any unsent video data. Notice though that the camera is never turned on and SlugCam consumes around 350mA during data transmission. The drop at the end shows

the WiFi module and the Pi being put to sleep and the Raspberry Pi turning to idle.

Another interesting feature to notice from the current consumption curves in Figure 11 is there is very little difference in power consumption when the camera is on and computer vision software is executed. This is the case without using the Pis GPU which we plan to use in the future. Another interesting observation when looking at the upper right graph comparing duty cycles B and F is that the camera consumes more power than the WiFi module. Comparing the curves in the bottom right graph for duty cycles D and E, we observe the difference in duty cycle duration; more specifically, duty cycle Es active period, i.e., when the system is not in idle state, is almost twice as long as duty cycle D. In this particular comparison, it shows the difference between storing the data locally (duty cycle D) and transmitting to a remote server (duty cycle E).

7.3 Adaptive Duty Cycling

In the next set of experiments, we have SlugCam cycle through different duty cycles as it would in normal operation. More specifically, we run the system for 5 minutes during which 3 distinct events occur. The first event is a false positive, e.g., light or heat radiation trigger the PIR, which in turn wakes up the camera and video is recorded and transmitted. The second event is data that should be recorded for reference later but does not get transmitted, because it is deemed not urgent, or the battery level is too low for transmission. The third and final event records video and transmits it to the remote server as it is considered by the user to be an urgent event.

In Figure 12 we show the events played over 5 minutes, while SlugCam uses different duty cycles in all three scenarios. The first scenario exemplifies a system with no adaptive abilities as baseline for comparison. In this scenario, all three events are handled by duty cycle E, as previously described, where data is recorded and transmitted for all events. The second scenario showcases the power saving benefits of adding computer vision to capture and analyze video footage. SlugCam is able

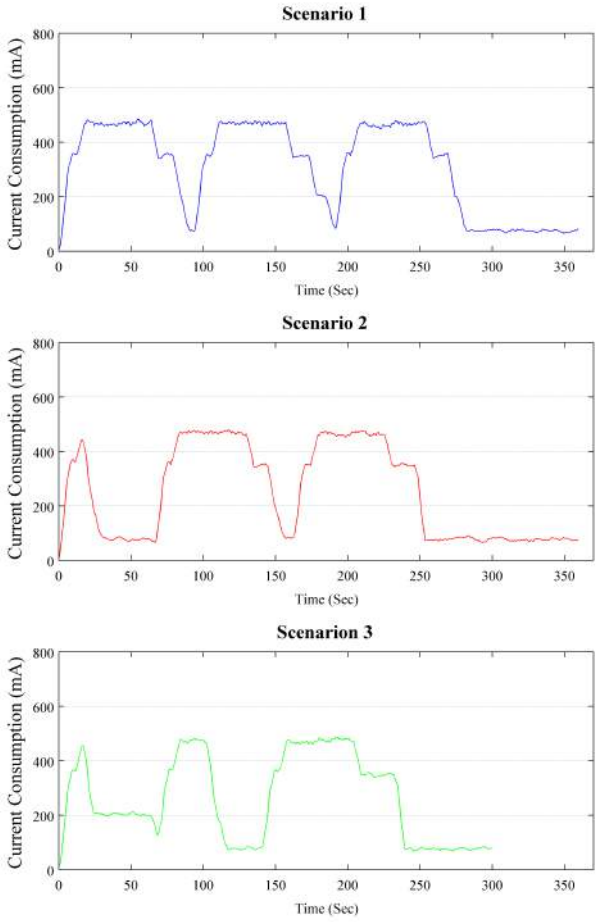

Figure 12: Current consumption data for the three application scenarios.

to correctly identify the first false positive event and immediately go back to low power state. This operation is represented by duty cycle C. In this scenario however, SlugCam then decides that the next two events should not only be recorded but immediately transmitted as well, which is handled by duty cycle E. Scenario 3 showcases the system ability to ignore the first event (duty cycle C), record and not transmit the second event (duty cycle D), and record and transmit the third event (duty cycle E).

8 Conclusion and Possible Future Improvements

8.1 Areas for Improvement

As was discussed a major set back for the system is its boot time and the ability to capture footage quickly. This can be improved with a lighter embedded Linux distribution or a move to a Real Time Operating system. Also to improve battery consumption costs Texas Instruments has recently release a more powerful and low-powered msp430 [26]. Along with improving battery consumption the system could then take advantage of more processing power for better frame rate performance when running computer vision algorithms. A quick fix to the more CPU performance would be to adopt the new Raspberry Pi 2 that features a 4 core system and a newer armv7 processor[14]. Heterogeneous processor systems are becoming more popular and SlugCam takes full advantage of this technique.

8.2 Conclusions Summary

Future iterations could include proper tracking hand offs in areas with cameras who capture overlapping footage to improve energy efficiency as in the MIMO paper [8]. Different wireless solutions could be easily integrated as more low powered WiFi modules are being released. Although the Node is susceptible to battery charge degradation, we do believe future iterations should make use of solid state relays, and SlugCam's mechanical relay is a risk of degradation as well. This thesis demonstrates the design, implementation, and evaluation of a outdoor solar-powered smart camera system. The combination of hardware and software features have not only met our application requirement goals of needing to deploy in outdoor environments on the UCSC campus, but have proven to be an efficient solution for many more application scenarios. With our efforts to document the entire system and open source all our software on a public repository, we hope SlugCam could be deem a cost effective powerful solution for computer vision researchers or other engineering fields. We also plan to deploy SlugCam nodes to assist the UCSC Informational Technology Services staff with student campus safety.

A SlugCam Custom Daughter Card

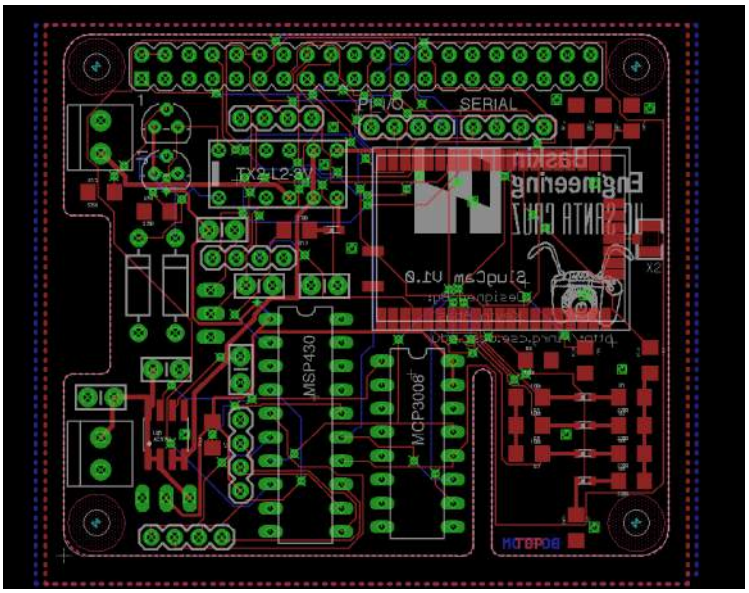

Figure 13: A screenshot of SlugCam's custom PCB board design.

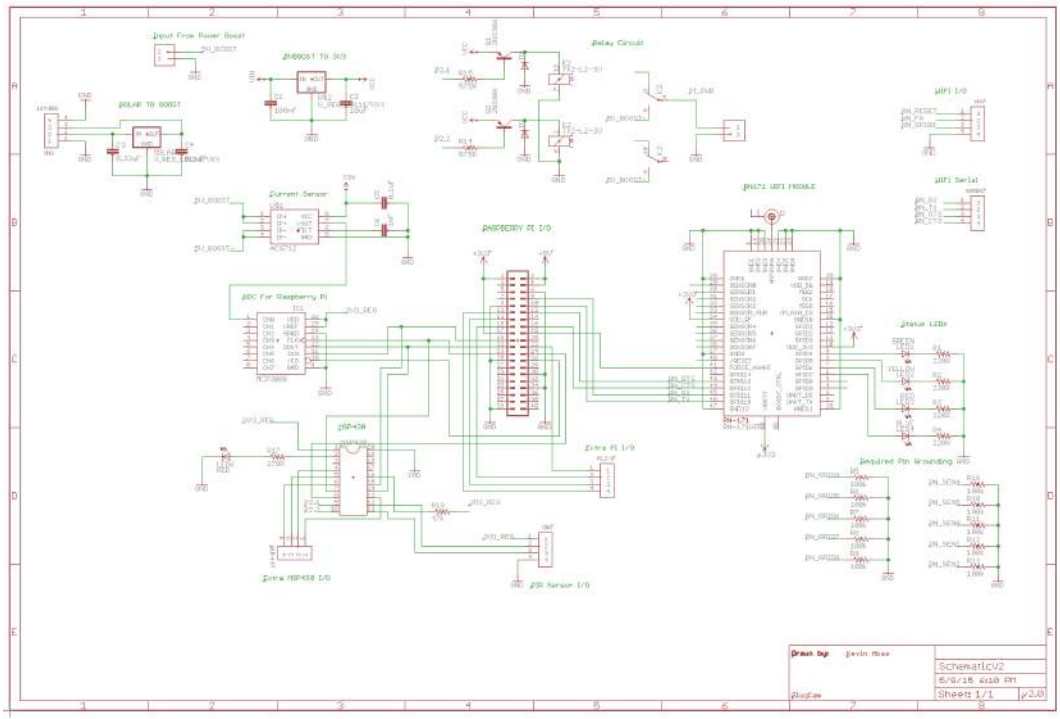

Figure 14: The Schematic of SlugCam's daughtercard.

B MSP430 Firmware Source

```
/*
 Slugcam MSP430 Firmware

 Date: 2/22/2015
 Author: Kevin Abas
 Last Modified: 6/2/2015
 File:

 This program allows the MSP430 the ability to monitor the passive
 infared sensor for motion, and to power on the SlugCam system using
 a dual coil relay. SlugCam also uses the MSP430 to then keep the SlugCam
 system awake by using an MSP430 on board timer(32Khz crystal). SlugCam
 communicates how long it wishes to stay awake using serial SPI
 communication. Should motion be a nusence, SlugCam also has the ability
 to tell the MSP430 to ignore the motion interupts for a longer period
 of time. Also using the msp430's available RTC clock library the msp430
 can also act as the systems clock by monitoring and syncing it.
 The MSP430 expects the following 10 byte packets when its slave select
 pin is active (note the \n is the deliminating LF):

    Set Time:
        'T000000\n__' (T+hrs(__)+min(__)+sec(__)+\n__)
    Set Date:
        'D00000000\n' (D+yr(____)+mon(__)+day(__)+\n)
    Set Sleep Countdown:
        'S000000\n__' (S+hrs(__)+min(__)+sec(__)+\n__)
    Special Commands:
        'CON\n______' (C+remainOn(ON)+\n______) <-Stay on forever
        'CDP0000\n__' (C+disable(__)+disableSec(__)+\n__) <-ignore PIR for
            this long
```

```
    *NOTE: The Pi should replace the trailing _'s with 0's

 GPIO Config:
   PIR Sensor:
     DOUT: pin 11 (P2.3
     EN: pin 12 (P2.4)
   SPI:
     MISO: pin 14 (P1.6) (UCB0SOMI)
     MOSI: pin 15 (P1.7) (UCB0SIMO)

     CE0: pin 6 (P1.4) (Slave Select)
     SCLK: pin 7 (P1.5) (UCB0CLK)

   Relay:
     Set: pin 9 (P2.1)
     Reset: pin 10 (P2.2)
 */
#include "msp430g2553.h"
#include <string.h>
#include <RTCplus.h> // Real-Time-Clock Library

//Interupt Service Routine Flags
volatile int motion_interupt_flag = LOW;
volatile int spi_interupt_flag  = LOW;
volatile int spi_error_flag     = LOW;
volatile int timer_interupt_flag = LOW;
volatile int timer_running_flag = LOW;
volatile int disable_pir_flag = LOW;

//Protected Timer Values
volatile unsigned int end_timer_seconds = 99;
volatile unsigned int end_timer_minutes = 99;
volatile unsigned int end_timer_hours = 99;
volatile unsigned int disable_pir_seconds = 99;
```

```
volatile unsigned int disable_pir_minutes = 99;


//Relay and PIR pins
const int relay_set_pin = 9;
const int relay_reset_pin = 10;
const int pir_dout_pin  = 11;
const int pir_enable_pin = 12;

//RTC Clock Data Structure
RealTimeClock my_clock;

//SPI RX Buffer Variables
char SPI_RX_buff[13];
volatile char SPI_TX_Error[16] = "Invalid\n";
char SPI_RX_index = 0;
volatile char SPI_TX_index = 0;
int  num_seconds, num_minutes, num_hours,
     num_year, num_month, num_day;
char *rx_seconds, *rx_minutes, *rx_hours,
     *rx_year, *rx_month, *rx_day;

/******************************
    Init Config Function
******************************/
void setup() {

  // initialize the pins for I/O
  pinMode(relay_set_pin, OUTPUT);
  pinMode(relay_reset_pin, OUTPUT);
  pinMode(pir_dout_pin, INPUT);
  pinMode(pir_enable_pin, OUTPUT);

  attachInterrupt(pir_dout_pin, motion_detected, RISING);
```

```
  digitalWrite(pir_enable_pin, HIGH);

  disable_wdt(); // Watch Dog Timer

  setup_spi();

  _BIS_SR(GIE); //Enable Global Interupts
  my_clock.begin(); //Start RTC

}


/*******************************
       Main Loop
*******************************/
void loop() {

  while (P1IN & BIT5);               // If clock sig from mstr stays low,
                                        // it is not yet in SPI mode
  flash_spi_detected();             // Blink 3 times

  if(timer_interupt_flag == HIGH){
    process_timer_expired();
    timer_interupt_flag = LOW;
  }else if(spi_interupt_flag == HIGH){
    process_spi();
  }else if(motion_interupt_flag == HIGH){
    motion_interupt_flag = LOW;
    process_motion();
  }else if( (motion_interupt_flag == LOW) &&
            (timer_interupt_flag == LOW) &&
              (spi_interupt_flag == LOW) ){
    __bis_SR_register(LPM3_bits + GIE);   // Enter LPM3, enable interrupts
  }
```

```
}

/*****************************
  Interupt Handling Functions
*****************************/
void process_motion(void){
  //If relay isn't on turn it on
  if(digitalRead(relay_set_pin) == LOW){
       timer_running_flag = HIGH;
       digitalWrite(relay_set_pin, HIGH);
       digitalWrite(relay_reset_pin, LOW);
  }
  set_sleep_timer(0, 5, 0); //auto sleep countdown is 5 min
  timer_running_flag  = HIGH;
  digitalWrite(pir_enable_pin, LOW); //disable PIR to save power
}

void process_spi(void){

  switch(SPI_RX_buff[0]){

     case 'T': //set RTC time
       get_buffer_time();
       my_clock.Set_Time(num_hours, num_minutes, num_seconds);

     case 'D': //set Date
       get_buffer_date();
       my_clock.Set_Date(num_year, num_month, num_day);

     case 'S': //set sleep countdown
       get_buffer_time();
       set_sleep_timer(num_seconds, num_minutes, num_hours);
       timer_running_flag  = HIGH;
```

```
        digitalWrite(pir_enable_pin, LOW);

      case 'C': //special command (stay on | diable PIR)
        if(SPI_RX_buff[1] == 'O') {
          digitalWrite(pir_enable_pin, LOW);
           end_timer_seconds = 99;
           end_timer_minutes = 99;
           end_timer_hours   = 99;
        }else if(SPI_RX_buff[1] == 'D') {
           get_disable_time();
           disable_pir_seconds = (my_clock.RTC_sec + num_seconds ) % 60;
           if( (end_timer_seconds <= 30 && ( num_seconds >= 30 ||
               my_clock.RTC_sec >= 30 ) )
               || ( num_seconds >= 30 && my_clock.RTC_sec >= 30 )){
             disable_pir_minutes = (my_clock.RTC_min + 1 + num_minutes) % 60;
           }else{
             disable_pir_minutes = (my_clock.RTC_min + num_minutes) % 60;
           }
           digitalWrite(pir_enable_pin, LOW);
           disable_pir_flag  = HIGH;
        }

       default:
        break;
    }
    SPI_RX_index = 0;
    spi_interupt_flag = LOW;
}

void process_timer_expired(void){
  //now turn off relay
    timer_interupt_flag = LOW;
  if(digitalRead(relay_set_pin) == HIGH){
```

```
      digitalWrite(relay_set_pin, LOW);
      digitalWrite(relay_reset_pin, HIGH);
  }
   timer_running_flag = LOW;
   end_timer_seconds = 99;
   end_timer_minutes = 99;
   end_timer_hours   = 99;
   if(disable_pir_flag == LOW){
     digitalWrite(pir_enable_pin, HIGH);
   }
}

/*****************************
    Interupt Service Routines
*****************************/

void motion_detected(void){
  motion_interupt_flag = HIGH;
}

//Using MSP430 native interupt notation
interrupt(TIMER1_A0_VECTOR) Tic_Tac(void) {
    my_clock++;          // Update time
    if( (my_clock.RTC_hr >= end_timer_hours) &&
          (my_clock.RTC_min >= end_timer_minutes) &&
            (my_clock.RTC_sec >= end_timer_seconds) ){
      timer_interupt_flag = HIGH;
       __bic_status_register_on_exit(LPM3_bits);
    }
   if((my_clock.RTC_min >= disable_pir_minutes) &&
            (my_clock.RTC_sec >= disable_pir_seconds) ){
      disable_pir_flag = LOW;
       digitalWrite(pir_enable_pin, HIGH);
       __bic_status_register_on_exit(LPM3_bits);
```

```
    }
}

//MSP430 SPI RX Service Routine
interrupt(USCIAB0RX_VECTOR) USCI0RX_ISR (void)
{
  if(IFG2 & UCB0RXIFG) {
    char value = UCB0RXBUF;
    if (value == '\n') { //ready to parse
        spi_interupt_flag = HIGH;
        __bic_status_register_on_exit(LPM3_bits);
    } else {
      SPI_RX_buff[SPI_RX_index] = value;
      SPI_RX_index++;
    }
  }
}

/*******************************
  Utility Functions
*******************************/
void disable_wdt(void){
  WDTCTL = WDTPW + WDTHOLD;// Stop watchdog timer
}

/* Incase of Hardware failures */
void fault_routine(void) {
  P1OUT &= ~BIT0; //turn off powerlight for warning
  while(1);
}

/* Delay For Debug LED function.*/
void led_delay(uint32_t d) {
  int i;
```

```
  for (i = 0; i<d; i++) {
    nop();
  }
}
/* SPI Debug LED flash */
void flash_spi_detected(void) {
    int i=0;
    for (i=0; i < 6; ++i) {
        led_delay(0x4fff);
        led_delay(0x4fff);
    }
}

void setup_spi(void) {
  P1DIR |= BIT0;                    //For debug, remove for deployment

  UCB0CTL1 = UCSWRST;               //Put msp430 USCI state machine in reset
  UCB0CTL0 = UCMODE_2|UCSYNC|UCCKPH; //4-pin, 8-bit SPI slave
  UCB0CTL0 |= UCMSB;                //Enable Most Singnificant bit first
  UCB0CTL0 &= ~UCMST;               //Set to slave.
  P1SEL = BIT4 + BIT5 + BIT6 + BIT7; //enable 4 SPI UCB0 pins
  P1SEL2 = BIT4 + BIT5 + BIT6 + BIT7;
  UCB0CTL1 &= ~UCSWRST;             //Set USCI state machine**
  IFG2 &= ~UCB0RXIFG;               //Clear pending recieve interupt
  IE2 |= UCB0RXIE;                  //Enable USCI0 RX interrupt

}

void set_sleep_timer(int seconds, int minutes, int hours) {
   end_timer_seconds = (my_clock.RTC_sec + seconds ) % 60;
   if( (end_timer_seconds <= 30 && ( seconds >= 30 || my_clock.RTC_sec >=
       30 ) )
        || ( seconds >= 30 && my_clock.RTC_sec >= 30 )){
     end_timer_minutes = (my_clock.RTC_min + 1 + minutes) % 60;
```

```
   }else{
     end_timer_minutes = (my_clock.RTC_min + minutes) % 60;
   }
   if( (end_timer_minutes <= 30 && ( minutes >= 30 || my_clock.RTC_min >=
       30 ) )
        || ( minutes >= 30 && my_clock.RTC_min >= 30 )){
     end_timer_hours = (my_clock.RTC_hr + 1 + hours) % 24;
   }else{
     end_timer_hours = (my_clock.RTC_hr + hours) % 24;
   }
}

void get_disable_time(void) {
   strncpy(rx_seconds, &SPI_RX_buff[3], 2);
   num_seconds = atoi(rx_seconds);
   strncpy(rx_minutes, &SPI_RX_buff[5], 2);
   num_minutes = atoi(rx_minutes);
}
void get_buffer_time(void) {
   strncpy(rx_seconds, &SPI_RX_buff[1], 2);
   num_seconds = atoi(rx_seconds);
   strncpy(rx_minutes, &SPI_RX_buff[3], 2);
   num_minutes = atoi(rx_minutes);
   strncpy(rx_hours, &SPI_RX_buff[5], 2);
   num_hours = atoi(rx_hours);
}
void get_buffer_date(void) {
   strncpy(rx_year, &SPI_RX_buff[1], 4);
   num_year = atoi(rx_year);
   strncpy(rx_month, &SPI_RX_buff[5], 2);
   num_month = atoi(rx_month);
   strncpy(rx_day, &SPI_RX_buff[7], 2);
   num_day = atoi(rx_day);
}
```

C Solar Harvesting Efficiency Explained

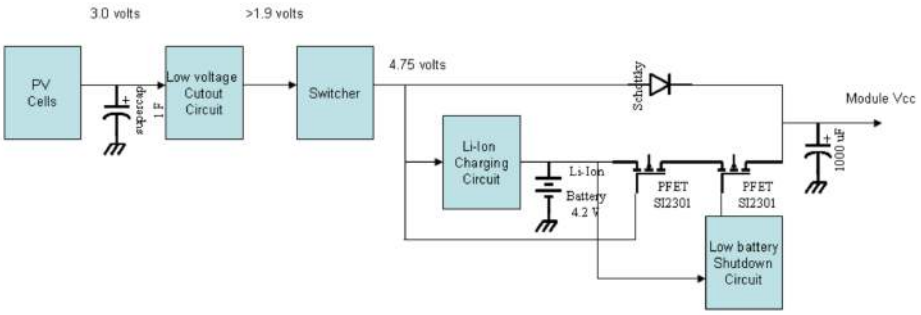

Figure 15: A basic High Level diagram of standard Solar energy harvesting circuits.[25]

Above in figure 15, one can see the most important components of an energy harvesting circuit. First, since solar radiation is extremely variable due to clouds or other objects flying over, it is crucial the system adapt to the panels constant change in voltage and current. This is why a large capacitor is placed before a low voltage cutout circuit. Then in this circuit using the PFET transistors the harvester is either charging the battery or powering the device if it is on.

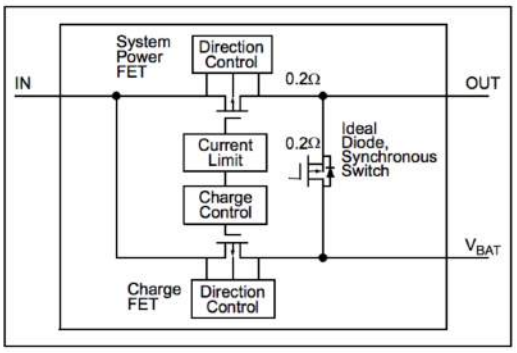

Figure 16: A basic High Level diagram of standard Solar energy harvesting circuits.[28]

The battery charging circuit we use from adafruit has a unique load sharing feature demonstrated in figure 16. With the panel providing 1 Amp in peak current and SlugCam normally using half of what is provided, we are able to have SlugCam running and charge the lithium-ion battery simultaneously.

References

[1] Kevin Abas, Caio Porto, and Katia Obraczka. Wireless smart camera networks for the surveillance of public spaces. *Computer*, 47(5):37–44, May 2014.

[2] J.K. Aggarwal and S. Park. Human motion: modeling and recognition of actions and interactions. In *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pages 640–647, Sept 2004.

[3] Andell Anees Alexander, Raymond Taylor, Vinujanan Vairavanathan, Yao Fu, Ekram Hossain, and Sima Noghanian. Solar-powered zigbee-based wireless motion surveillance: a prototype development and experimental results. *Wireless Communications and Mobile Computing*, 8(10):1255–1276, 2008.

[4] Z-Wave Alliance. The wireless z-wave communications standard official website, 2014. `http://z-wavealliance.org/about_z-wave_technology/`.

[5] Inc. Bluetooth SIG. The wireless z-wave standard official website, 2014. `https://developer.bluetooth.org/TechnologyOverview/Pages/BLE.aspx`.

[6] Lorena Calavia, Carlos Baladrón, Javier M Aguiar, Belén Carro, and Antonio Sánchez-Esguevillas. A semantic autonomous video surveillance system for dense camera networks in smart cities. *Sensors*, 12(8):10407–10429, 2012.

[7] Phoebus Chen, Kirak Hong, Nikhil Naikal, S Shankar Sastry, Doug Tygar, Posu Yan, Allen Y Yang, Lung-Chung Chang, Leon Lin, Simon Wang, et al. A low-bandwidth camera sensor platform with applications in smart camera networks. *ACM Transactions on Sensor Networks (TOSN)*, 9(2):21, 2013.

[8] Zichong Chen, G. Barrenetxea, and M. Vetterli. Share risk and energy: Sampling and communication strategies for multi-camera wireless monitoring networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 1862–1870, March 2012.

[9] Zichong Chen, P. Prandoni, G. Barrenetxea, and M. Vetterli. Poster abstract: Sensorcam: An energy-efficient smart wireless camera for environmen-

tal monitoring. In *Information Processing in Sensor Networks (IPSN), 2012 ACM/IEEE 11th International Conference on*, pages 99–100, April 2012.

[10] Mat Dirjish. Whats the difference between thin-film and crystalline-silicon solar panels, May 2012. `http://electronicdesign.com/power-sources/`.

[11] Prabal Dutta, Jonathan Hui, Jaein Jeong, Sukun Kim, Cory Sharp, Jay Taneja, Gilman Tolle, Kamin Whitehouse, and David Culler. Trio: Enabling sustainable and scalable outdoor wireless sensor network deployments. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, IPSN '06, pages 407–415, New York, NY, USA, 2006. ACM.

[12] Justin Eure. Scientists pinpoint the creeping nanocrystals behind lithium-ion battery degradation. *Research & Development*, May 2014. http://www.rdmag.com/news/.

[13] Jorge Fernandez-Berni, Ricardo Carmona-Galán, Gustavo Linán-Cembrano, Ákos Zarándy, and A Rodriguez-Vazquez. Wi-flip: A wireless smart camera based on a focal-plane low-power image processor. In *Distributed Smart Cameras (ICDSC), 2011 Fifth ACM/IEEE International Conference on*, pages 1–6. IEEE, 2011.

[14] Raspberry Pi Foundation. Raspberry pi model b 2, 2010. `https://www.raspberrypi.org/products/raspberry-pi-2-model-b/`.

[15] Raspberry Pi Foundation. The raspberry pi camera module, 2013. `http://elinux.org/Rpi_Camera_Module`.

[16] Raspberry Pi Foundation. The raspberry pi model b+, February 2015. `https://www.raspberrypi.org/products/model-b-plus/`.

[17] L. Gasparini, R. Manduchi, M. Gottardi, and D. Petri. An ultralow-power wireless camera node: Development and performance analysis. *Instrumentation and Measurement, IEEE Transactions on*, 60(12):3824–3832, Dec 2011.

[18] 6LowPAN Working Group. The wireless z-wave standard official website, 2014. http://datatracker.ietf.org/wg/6lowpan/charter/.

[19] Arch Linux Group. The arch linux operating system for the raspberry pi, 2014. https://www.archlinux.org/.

[20] Carl Hartung, Richard Han, Carl Seielstad, and Saxon Holbrook. Firewxnet: A multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services*, MobiSys '06, pages 28–41, New York, NY, USA, 2006. ACM.

[21] Yang He, Daniela Molina Piper, Meng Gu, Jonathan J Travis, Steven M George, Se-Hee Lee, Arda Genc, Lee Pullan, Jun Liu, Scott X Mao, et al. In situ transmission electron microscopy probing of native oxide and artificial layers on silicon nanoparticles for lithium ion batteries. *ACS nano*, 8(11):11816–11823, 2014.

[22] Stephan Hengstler, Daniel Prashanth, Sufen Fong, and Hamid Aghajan. Mesheye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 360–369. IEEE, 2007.

[23] Weiming Hu, Tieniu Tan, Liang Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352, Aug 2004.

[24] Microchip Incorporated. Ultra-low powered embedded wifi module, 2014. Available at http://www.microchip.com/rn171.

[25] RF Monolithics Incorporated. A solar energy harvester for wireless sensor networks, 2010. http://www.mouser.com/pdfdocs/mesh-rfm-solar-wireless-sensor-networks.pdf.

[26] Texas Instruments Incorporated. Msp432p401x datasheet, 2010. http://www.ti.com/lit/ds/slas826a/slas826a.pdf.

[27] T.I. Incorporated. Msp430g2x53 datasheet, 2013. http://www.ti.com/lit/ds/symlink/msp430g2213.pdf.

[28] Adafruit Industries. Li-ion/li-polymer battery charge management controller, 2014. https://www.adafruit.com/products/390.

[29] Itseez. Open source computer vision software, 2015. http://www.opencv.org.

[30] Swantje Johnsen and Ashley Tews. Real-time object tracking and classification using a static camera. In *Proceedings of IEEE International Conference on Robotics and Automation, workshop on People Detection and Tracking*, 2009.

[31] Arvind Kandhalu, Anthony Rowe, Ragunathan Rajkumar, Chingchun Huang, and Chao-Chun Yeh. Real-time video surveillance over ieee 802.11 mesh networks. In *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*, pages 205–214. IEEE, 2009.

[32] Minsoo Kim, Chong-Min Kyung, and Kang Yi. An energy management scheme for solar-powered wireless visual sensor networks toward uninterrupted operations. In *SoC Design Conference (ISOCC), 2013 International*, pages 023–026. IEEE, 2013.

[33] Peter Korsgaard. Embedded linux systems image generator, 2014. Available at http://buildroot.uclibc.org/.

[34] Erik Ljung, Erik Simmons, Alexander Danilin, Richard Kleihorst, and Ben Schueler. 802.15. 4 powered distributed wireless smart camera network. In *Workshop on Distributed Smart Cameras (DCS 2006), Boulder CO*, 2006.

[35] M. Magno, D. Brunelli, P. Zappi, and L. Benini. A solar-powered video sensor node for energy efficient multimodal surveillance. In *Digital System Design Architectures, Methods and Tools, 2008. DSD '08. 11th EUROMICRO Conference on*, pages 512–519, Sept 2008.

[36] M. Magno, F. Tombari, D. Brunelli, L. Di Stefano, and L. Benini. Multimodal video analysis on self-powered resource-limited wireless smart camera. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 3(2):223–235, June 2013.

[37] Allegro Microsystems. Acs712 hall effect current sensor, 2014. `http://www.allegromicro.com/`.

[38] Leland Miller, Kevin Abas, and Katia Obraczka. Scmesh: Solar-powered wireless smart camera mesh network. *Internation Conference Computer Communication and Networks (ICCCN)*, 2015. (Under Review).

[39] N.R.E.L. National renewable energy laboratory research data, 2014. `http://www.nrel.gov/solar_radiation/`.

[40] K. Obraczka, R. Manduchi, and J.J. Garcia-Luna-Aveces. Managing the information flow in visual sensor networks. In *Wireless Personal Multimedia Communications, 2002. The 5th International Symposium on*, volume 3, pages 1177–1181 vol.3, Oct 2002.

[41] Panasonic. Ble panasonic soc module, 2015. `http://eu.industrial.panasonic.com/products/wireless-connectivity/`.

[42] Andreas Papalambrou, Panagiotis Soufrilas, Artemios G Voyiatzis, and Dimitrios N Serpanos. A secure dtn-based smart camera surveillance system. In *Proceedings of the Workshop on Embedded Systems Security*, page 3. ACM, 2011.

[43] Alvaro Pinto, Zhe Zhang, Xin Dong, Senem Velipasalar, Mehmet Can Vuran, and Mustafa Cenk Gursoy. Analysis of the accuracy-latency-energy tradeoff for wireless embedded camera networks. In *Wireless Communications and Networking Conference (WCNC), 2011 IEEE*, pages 2101–2106. IEEE, 2011.

[44] Umakishore Ramachandran, K. Hong, L. Iftode, R. Jain, R. Kumar, K. Rothermel, Junsuk Shin, and R. Sivakumar. Large-scale situation awareness with cam-

era networks and multimodal sensing. *Proceedings of the IEEE*, 100(4):878–892, April 2012.

[45] Hele Savin, Pivikki Repo, Guillaume von Gastrow, Pablo Ortega, Eric Calle, Moises Garn, and Ramon Alcubilla. Black silicon solar cells with interdigitated back-contacts achieve 22.1% efficiency. *Nature Nanotechnology*, advance online publication, May 2015.

[46] Adolph Seema and Martin Reisslein. Towards efficient wireless video sensor networks: A survey of existing node architectures and proposal for a flexi-wvsnp design. *Communications Surveys & Tutorials, IEEE*, 13(3):462–486, 2011.

[47] W Richard Stevens, Bill Fenner, and Andrew M Rudoff. *UNIX network programming*, volume 1. Addison-Wesley Professional, 2004.

[48] M.K. Stojcev, M.R. Kosanovic, and L.R. Golubovic. Power management and energy harvesting techniques for wireless sensor nodes. In *Telecommunication in Modern Satellite, Cable, and Broadcasting Services, 2009. TELSIKS '09. 9th International Conference on*, pages 65–72, Oct 2009.

[49] Sujesha Sudevalayam and Purushottam Kulkarni. Energy harvesting sensor nodes: Survey and implications. *Communications Surveys & Tutorials, IEEE*, 13(3):443–461, 2011.

[50] Thiago Teixeira, Dimitrios Lymberopoulos, Eugenio Culurciello, Yiannis Aloimonos, and Andreas Savvides. A lightweight camera sensor network operating on symbolic information. In *Proceedings of the 1st Workshop on Distributed Smart Cameras*, 2006.

[51] ZigbeeAlliance. The wireless zigbee standard specifications, 2014. `http://zigbee.org/zigbee-for-developers/network-specifications/`.