*DIY Motion Sensor Camera*

Tony Phan  
*Computer Science*  
*Colorado State University*Littleton, United States

tonyphan@colostate.edu

Beau Steinle  
*Computer Science*  
*Colorado State University*Fort Collins, United States  
bsteinle@colostate.edu

*Abstract*—Home security is a concern for homeowners and renters alike, but costs for a full security system can be prohibitively high. Instead of a comprehensive home security system, consumers can install motion sensing cameras on major entry points and high traffic areas. These cameras would then be able to record the coming and going of visitors when the owner or renter is not home. This project was developed to test the feasibility of creating these types of cameras as an affordable DIY project, while still having capabilities of similar products found on the market today. The criteria used were detecting motion of approaching visitor, recording the visitor when motion was detected, and uploading a video to a remote service for the owner to view from anywhere. With these criteria we developed a prototype motion sensing camera using a raspberry pi board, passive infrared sensor, and a small camera.

Keywords—Motion Sensor, Camera, Raspberry Pi, Home Security, DIY.

# Introduction

Our project was to develop and evaluate a system using a single-board computer. With this objective in mind, we decided to build a home security system using a raspberry pi board and operating system with Wi-Fi capabilities. We then decided to attach a motion detector and a camera onto the board to record a video when motion was detected. To link both the camera and the motion detector together, we explored and developed software to do so and integrated a feature of sending the videos to a Dropbox account. Along the way, we discovered how to convert video files from an “.h264” file to an “.mp4” file and explored different ways of integrating threading into the software. The final product was a home security box that recorded 20 second videos when motion was detected and sent the video to a Dropbox account where a person can view the videos from any device.

# Problem Characterization

As the world moves towards a more technological society, online shopping has become more popular. As more people begin shopping online, there will be more packages delivered to homes around the world. Consequently, however, there will also be an increase in the number of porch thieves, thus increasing the need for security systems for every house around the world.

# Proposed Solution

A motion sensing camera that uploads recorded videos in real-time would enable concerned individuals to monitor the inside or outside of their homes. They would simply mount the camera pointed in the direction they want to watch and then they can view any recorded videos on a separate device.

# Methodology

The device needed to include features that made it useful and convenient to use. A camera that continually recorded to physical media and required a user to manually remove a storage device to look through all of the contents to find what they’re looking for was inconvenient and cumbersome. So, our team developed the following criteria for this project as the minimum viable product (MVP):

* Device continuously monitors area for motion
* Device can withstand being outside for long periods of time
* When motion is detected, camera turns on to record a video
* Camera stays off when not in use
* Videos are uploaded to a remote storage service and then deleted from local storage
* Uploading videos needed to occur immediately after completing recording
* Device immediately resumes scanning for motion after a video is done recording

# HardWare

To meet our MVP criteria, we needed to select the major hardware used to develop the device prototype:

* *Raspberry Pi 3B+ Single-board Computer*: The Raspberry Pi 3B+ was selected for its ease of use, thorough documentation, SD card slot, and Wi-Fi capabilities.
* *Canakit 5V 2.5A Raspberry Pi 3B+ Power Supply/Adapter*: This was an affordable power source choice that was also compatible with our chosen board. A wired power source was chosen to reduce costs of the prototype device.
* *EIKS**3pcs PIR Motion Sensor*: For the motion sensor, we chose this particular model because it used a passive infrared detector, came with mounting brackets (for cleaner and easier assembly), and had adjustable sensitivity and time delay components.
* *Raspberry Pi Camera Module, 5 Megapixel 1080p Sensor*: As for the camera, we found this model that came with the proper cable to connect to the Raspberry Pi board, and it came with a mountable acrylic case.
* *Hard Plastic Case*: Chosen to house the electronic components. The case we decided on included a watertight sealable lid.

The cost of these parts is seen as the largest barrier for this device to be viable as a DIY home security solution. The total cost of the prototype device was $77.83, which when compared to devices on the market is a bit overpriced. The device we built was developed as a prototype and we focused on ease of development by using well documented components. A more developed version of this device could reduce overall cost by replacing components with the cheapest, minimum requirements needed.

Assembly of the device was straight forward as the Raspberry Pi board is well documented and our chosen components were all compatible directly with the board. The camera, motion sensor, and the power source plugged directly into the board and holes were then cut into the hard plastic case for the camera and the motion sensor to look through, as well as a small hole for the power cable to be run through. Mounting the components into the box was accomplished simply using tape, as this is a prototype device and ease of disassembly was needed to quickly change configurations as we developed the project.

# Software

The only software that needed to be chosen and installed externally onto the board through the SD card was the operating system. For this we went with the standard Raspberry Pi Operating System for compatibility and ease of use. With the hardware set up and board running, we were ready to start developing the code to control our camera. To develop our project, we decided to create a python script to control the motion sensor camera. This decision was based on the third-party libraries that were already available to download in python for controlling our board, camera, and for uploading the video after recording.

# Libaries Used (third-party packages)

There were several libraries available for download that helped in developing the device:

* *RPi.GPIO*: This library provided functions for controlling and monitoring the Raspberry Pi board. With this we were able to monitor the input pin that the motion sensor was plugged into and detect a motion event to trigger the recordings.
* *picamera*: This library provided functions for controlling our camera module. These functions allowed us to start/stop recordings and control the format/quality of produced recordings.
* *MP4Box*: Recordings from picamera did not include an .mp4 format that is most common for videos today. As we wanted these videos to be viewable from the widest range of devices, so we installed this library to convert from the .h264 format to .mp4. This library is in the form of a bash command and is not written in python.
* *Dropbox*: This provided functions allowed an easy way to upload videos to a user’s personal Dropbox account. There were a few set-up steps that needed to happen on the user’s Dropbox account for the videos to be sent. Namely, a user would have to create a folder for the videos and generate an access key that would have to be put onto the board’s local memory.

# Development Process

Even with these libraries helping develop our code, there were several challenges that needed to be addressed in our program. For our original implementation we developed a simple algorithm: poll the motion sensor’s input pin using a loop, sleep the program for 0.5 seconds and if no input then it loops back around, and call recording function if input received. If the recording function was called it would initialize the camera, start the recording with a 30 second timeout, stop the recording, then move on to convert to .mp4, and finally upload onto Dropbox.

This implementation did work on a very basic level, but we ran into two major issues: false positive motion inputs and latency from finishing a recording to starting detection again.

Our high false positive rate was resulting in several useless videos with no one in them to be recorded. This could be due in part to the sensor that we purchased being somewhat cheaper, but we wanted to optimize this as best we could in our code. For this we revisited how we monitored for input and triggered a recording to start. Using the RPi.GPIO library, we were able to move away from our original polling method and establish an event. Instead of polling the pin every half second, we could suspend the main process until it received input, triggering an event and falling through to our function. This did reduce our false positive rate, but we were still recording several empty scenes. So, we decided to implement a second motion check. After a motion event occurred, the program would fall through and then sleep for 0.3 seconds, then we would poll the pin to see if it was still receiving a motion signal. If the second check passed, we would then trigger the recording process. Using this method, we were able to completely eliminate our false positive issue. The 0.3 second sleep duration was the result of thorough trials and was the minimum amount of time that didn’t still trigger false positives.

Next, we addressed the latency between recording a video and starting detection. Once a video had completed its recording, the process would then convert the file to .mp4 format and upload it to Dropbox. These extra steps took a significant amount of time before the process would return to the main program to listen for more motion events. To eliminate this issue, we decided to introduce threading to our program. Once a video was finished recording, a thread would be spawned that handled the video conversion as well as the uploading, while the main process would resume listening for motion events. This did almost eliminate the latency problem but resulted in another issue to be addressed.

With 30-second-long videos being recorded each time with default resolution and quality, the amount of time for the conversion/upload thread to complete was too long. On average it would take almost a minute before completion, during which time another video could have completed its recording and was attempting to create another thread for its own conversion. As the Raspberry Pi 3B+ board only provides up to 4 threads at any time, this was causing hang-ups and strange behavior with our device. Occasionally, the event detection would be caught in an infinite loop, starting a recording immediately after returning, or the program would freeze up in a deadlock. We first thought to set flags to ensure it was safe to create a new thread, but this would have also introduced large time gaps between videos. So, we then decided to experiment with the length, quality, and resolution of our videos. The original recordings defaulted to a high-quality 1080p recording and at 30 seconds long, it was not surprising the conversion and upload would take a minute. But we couldn’t just drop the quality and length to the minimum, a 5 second grainy video would not be useful for a user. To balance quality and speed, we experimented with several different configurations. In the end we were able to achieve a reasonably quick conversion/upload time with a watchable quality using: 640x480 resolution, mid-level quality, and a 20 second video.

With this implementation we were able to achieve the desired results for our MVP: a continuously motion sensing device that starts the camera and records video when a motion event occurs, then converts the video to .mp4, uploads the video to a remote location, and has minimal delay between videos.

# Conclusion

Through exploring and developing a DIY security system using a single-board system, we were able to replicate and create our own simplified system. Although we encountered many issues during the development process, we eventually arrived at a fully functional motion detecting camera that would record videos when motion was detected and upload them to a remote storage for a user to view. With the final device functioning as intended, we believe that creating these as a DIY security solution is highly viable. The largest barriers, however, would be the cost of hardware and the background knowledge needed to create the device. For the costs, we believe it could be reduced a bit by using less expensive components (since the components we used exceeded what was necessary). Likewise, being the first prototype, we were able to gain the necessary knowledge to replicate and reproduce the device in a much more efficient manner.

##### References

1. API - picamera.camera Module. 10. API - picamera.camera Module - Picamera 1.10 documentation. https://picamera.readthedocs.io/en/release-1.10/api\_camera.html. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
2. Aktar, K. (n.d.). How to convert h264 to mp4 with MP4Box on Raspberry Pi in Python. TheAmplituhedron. <https://www.theamplituhedron.com/articles/How-to-convert-h264-to-mp4-with-MP4Box-on-Raspberry-Pi-in-Python/>.
3. Basic Recipes. Basic Recipes - Picamera 1.10 documentation. <https://picamera.readthedocs.io/en/release-1.10/recipes1.html>.
4. Broadcom BCM2837 Specs. GadgetVersus. https://gadgetversus.com/processor/broadcom-bcm2837-specs/.
5. Hallinan, D. (2019, October 22). Monday Mentorship: Using Python to Upload a File to Dropbox. Kloudless. https://kloudless.com/blog/monday-mentorship-using-python-to-upload-a-file-to-dropbox/
6. Introduction. Getting Started With the Camera Module. https://projects.raspberrypi.org/en/projects/getting-started-with-picamera.
7. Muller, D. (2020, July 30). How To Use subprocess to Run External Programs in Python 3. DigitalOcean. https://www.digitalocean.com/community/tutorials/how-to-use-subprocess-to-run-external-programs-in-python-3.
8. Sanjeev, A. (2018, April 23). How to Interface a PIR Motion Sensor With Raspberry Pi GPIO: Raspberry Pi. Maker Pro. https://maker.pro/raspberry-pi/tutorial/how-to-interface-a-pir-motion-sensor-with-raspberry-pi-gpio.