Smart Safe

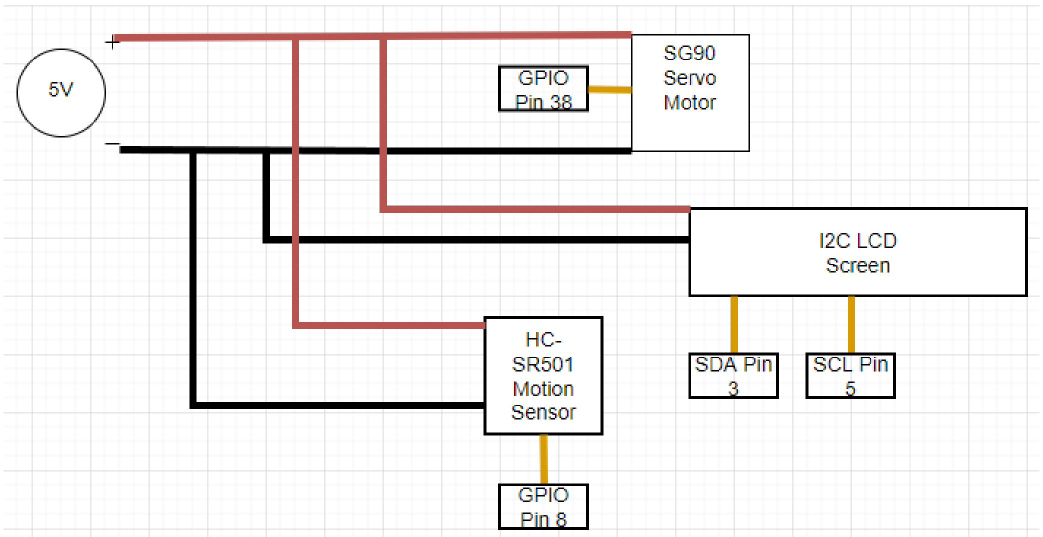
CSCI 43300 Introduction to Internet of Things

Chip Simmerman
csimmer@iu.edu

Samip Vaidh

December 10, 2021

Node Setup, Circuit Design, & Description:



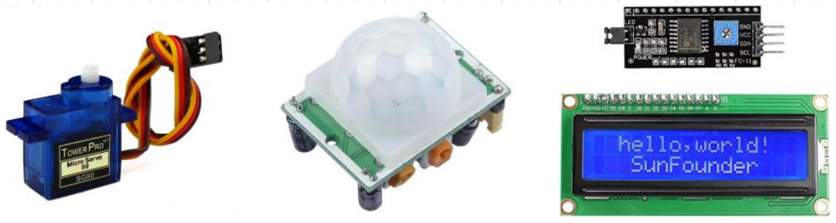


Figure 1: This diagram shows the sensors that all use 5V. The first item is the servo motor (actuator) which uses PWM from pin 38 to rotate its arm in a circular fashion. The second item is the LCD screen which has two important connections to the screen; the SDA (data) and SCL (clock) pins which connect directly to the SDA and SCL pins on the Raspberry Pi. Additionally, the LCD screen also has a control that can be turned to increase or decrease the contrast. The final sensor in this picture is the HC-SR501 Motion Sensor which has a data pin that connects to GPIO pin 8 on the Pi. Like the LCD screen, this motion sensor has two controls to change the distance and time delay of the sensor. In our use, we set it to the minimum distance of 3 meters and the lowest time delay.

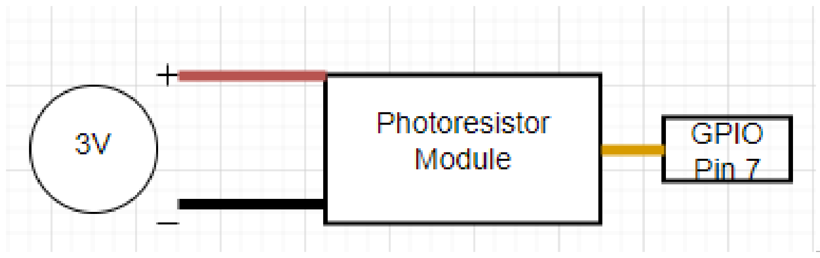
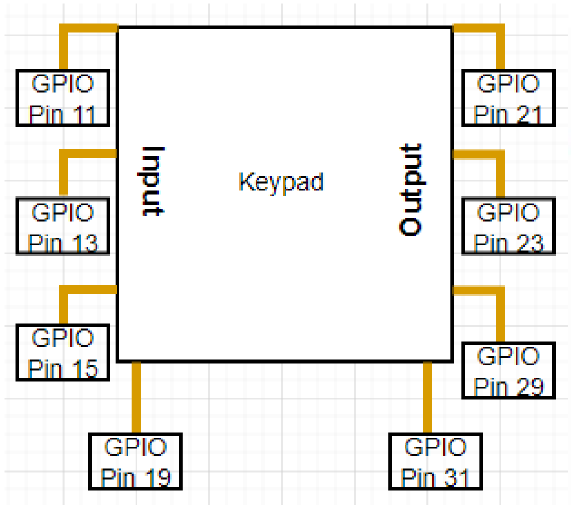


Figure 2: This diagram shows the photoresistor sensor module. The power is supplied from the 3V pin of the Pi to the sensor (red). The main wire is connected to GPIO pin 7 to read and manipulate data fed into the sensor. This module also has LEDs to show when it receives

power and when light is detected. Additionally, the module includes a control that can change the sensitivity of the sensor.



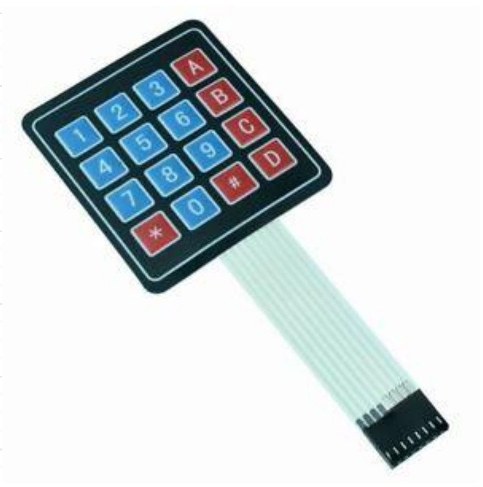
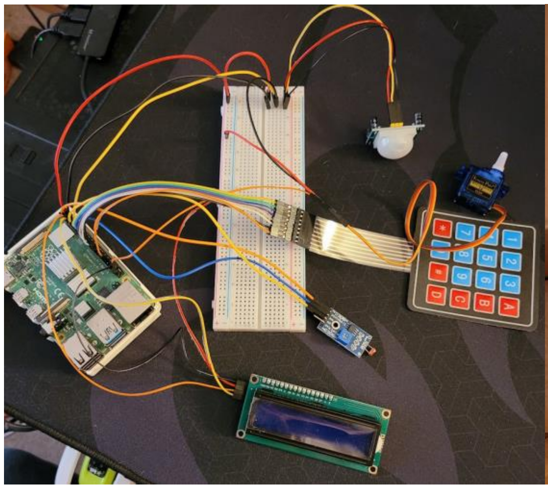
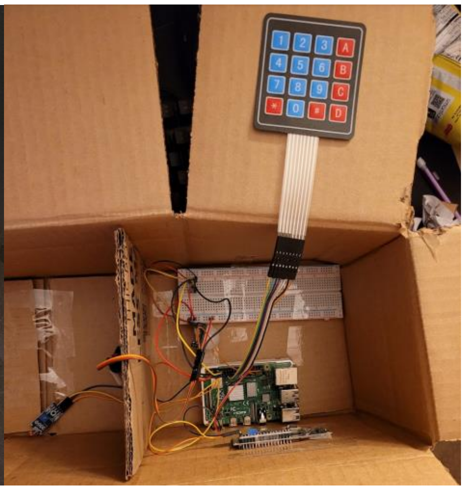


Figure 3: This diagram shows the final sensor which is the keypad. On the left the 4 pins, GPIO pin 11, GPIO pin 13, GPIO pin 15, and GPIO pin 19 are the inputs of the keypad array. The right half is the output of the keypad array and the pins here are respectively GPIO pin 21, GPIO pin 23, GPIO pin 29, and GPIO pin 31. This sensor works by initially sending a pulse to each of the 4 input pins. When a number is pressed, a connection is formed which allows the pulse to travel to one of the 4 output pins. The cross-section formed by the 4 input and 4 output pins allows any of the 16 buttons to be detected when pressed.

Sensors Combined:

Figures 4 & 5: These images show the initial stages of putting our physical project components together. The left image shows our system with just the Raspberry Pi, breadboard, and sensors/actuators. The right image shows the next step of getting all of the hardware positioned nicely into a box that will act as our safe. We sectioned off most of the components for a cleaner look and used the other side of the divider as the container for the safe.

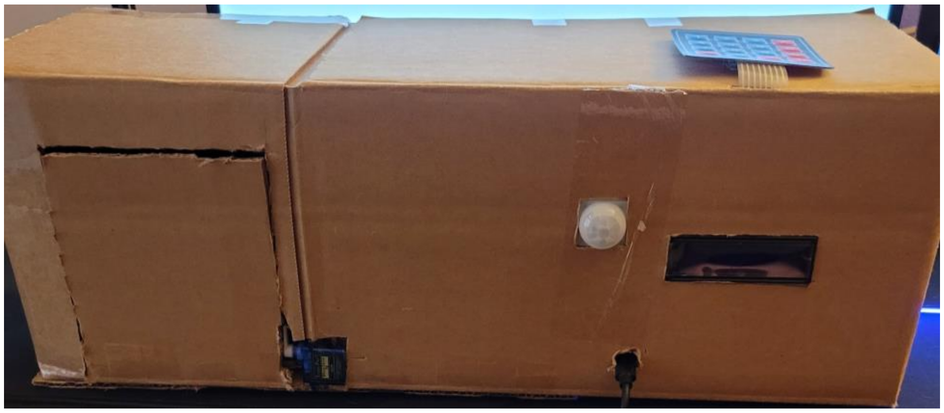
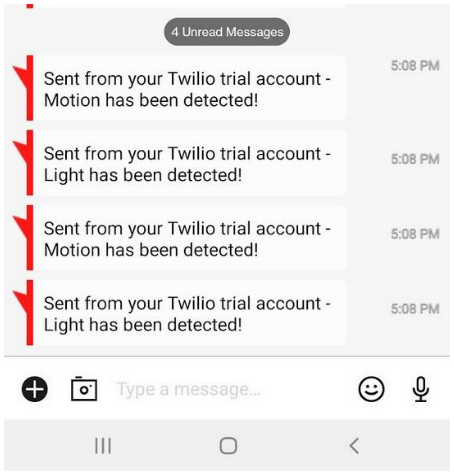


Figure 6: This image shows the final box that represents the shape of the system. The smart safe has visible pieces of sensors on it. On the very top is the keypad for the user input. On the middle right side is the LCD screen to display messages. To the left side of the LCD display sits the motion detector. The cable at the bottom is the power supply to the Raspberry Pi which is housed inside the box. Additionally, the light sensor sits behind the door of the safe (not pictured). The final component that opens the cut-out door on the left side of the box is the actuator/servo motor.



4 Unread Messages

Sent from your Twilio trial account - Motion has been detected! 5:08 PM

Sent from your Twilio trial account - Light has been detected! 5:08 PM

Sent from your Twilio trial account - Motion has been detected! 5:08 PM

Sent from your Twilio trial account - Light has been detected! 5:08 PM

Figure 7: Here is the image when the program sends a message using Twilio to the user's phone every 20 seconds when motion or light is detected. The program is always searching for motion as well as light but is told to only send the message every 20 seconds. Therefore, the user is not spammed with text messages.

Application Code Development For Sensors:

At the beginning of the project, we wanted to simply get each of our components to work separately using Python before combining them into the final product. First, we got the HC-SR501 Motion Sensor to function alone by simply turning on an LED and printing out a message to the console when motion was detected, and then turning the LED off after a period of time passed. This was easy to do and demonstrated that the sensor outputs a high voltage when motion is detected. Next, we had the Photoresistor Light Sensor work similar to the motion sensor where we continuously checked the sensor's input.

An interesting discovery we made was that when light was detected, the input given by the sensor was 0 rather than 1. This is because the resistance of the photoresistor increases when light is detected. Next in line was getting the SG90 Servo Motor to function properly. This actuator was difficult to understand initially due to its use of PWM to set the position of the motor's arm. To initially demonstrate its functionality, we used a program that looped through duty cycles from 2.5 to 13, which allowed the motor to rotate its arm before resetting. The fourth component we utilized was the I2C LCD screen that came with a Raspberry Pi starter kit. Luckily, our LCD screen came with a backpack component that required only 3 pins to be used rather than the several required without the backpack. This screen proved to be very difficult when trying to get it to display information, so we used a helper Python program from Circuit Basics that allowed us to display and clear strings rather easily. After getting the LCD screen configured with the correct code and contrast, we set up our last hardware component: the keypad. The keypad we used requires 8 GPIO pins to be connected, which took up a lot of space on our Raspberry Pi. Fortunately, this sensor does not require grounding or a power supply. We used a program that continuously loops through each input pin to check for a button being pressed. When a button is pressed, the corresponding character is then printed from an array.

Lastly, we wanted to introduce functionality into the project beyond sensors and actuators. Therefore, we utilized the services of Twilio to send SMS messages through Python when directed. Twilio offers a free trial in which one can register to use one of its phones. Once given the authentication token and account ID, a user can then utilize the Twilio Python package to send text messages to a phone number of their choice. It was incredibly easy to set up and allowed us to take our smart safe device to a new level of communication.

After having each of our hardware and software components working separately, we still had to connect them into one product through software to behave like a safe. To do so, we started by building the functionality of having a code that a user has to enter through the keypad and have the program react accordingly to the correctness of the code. After this step was completed, we incorporated the LCD screen into the system to display information about the safe, such as the code being entered, when the code is incorrect, when the safe is opening, and when the safe is closing. Third, we added the servo motor to react when a code is entered correctly in order to open and close the safe. Next, we added the light and motion sensors to continuously detect light and motion while the safe is on. We quickly found that with reading input from the keypad, displaying info on the LCD screen, and controlling the servo motor that there was too much going on to consistently get sensor readings. Therefore, we had to use separate threads using the RepeatedTimer class we found through Stack Overflow to call functions every 20 seconds for sensor readings. Having these sensors running on their own threads and having them call their functions on a timer allowed us to introduce the final part of our system which was the Twilio client. We opted to include the messaging functionality last due to constantly debugging the program and not wanting to waste any of our free trial messages. Once all of these components were combined into one program, the safe was complete and functioned as expected.

Application Explained:

The idea of the project is to design a safe system that will detect motion as well as light when the safe is open. Starting from the first input, which is the keypad array, the user will input a code that will be checked for correctness. If the code does not match, the LCD screen will display an error message stating the input is incorrect before resetting the code. If the user enters the correct password the safe will then tell the servo motor (actuator) to open the door. Once the door is open the user can then lock the safe by typing in the same password. The program will then report a message on the LCD display explaining that the safe is closing.

The motion sensor and light sensor are also here working as well. In the sense that once there is some motion detected the program will alert the user via a text message every 20 seconds if there is any new movement. The message will show on their mobile device as “Motion has been detected.” The light sensor is used inside the safe. So whenever the safe is open and there is light cast upon the photoresistor the signal will then tell the program to report a message to the user via a text message stating that there has been light detected. Meaning that the safe door is currently open.

Issues Encountered/Resolved:

Prior to settling on this project idea and getting permission, we originally wanted to create a smart home assistant with the Raspberry Pi using the open-source Google Assistant SDK to read from connected sensors. This quickly proved to be a problematic project idea as we found that Google removed the “hotword” functionality allowing a user to query the assistant at any time. Furthermore, we found that the software was severely lacking in “smart capabilities” and we would have to program any sensor reading to voice communication we needed. After realizing that our original idea was outside of the scope for this course, we decided to settle on this smart safe.

Most of the issues we encountered were relatively easy to overcome in this project. For example, we found that we could not use the light and motion sensor by themselves as the rest of the program would delay the sensor output. Unfortunately, traditional threading would not work for our purposes of continuously sensing and calling functions, so we had to implement the RepeatedTimer class to do so. We also found that the SG90 mini servo motor twitches when connected to the Pi, but we were unable to resolve this issue as we learned the behavior was normal. Chip, who worked on putting the Pi and hardware components together into a container, found it rather difficult to find the best-sized box for the project. After finding the right box, he had to resize it further so that the wires connected to the sensors and the Pi could adequately reach the edges of the box. Working with cardboard was fairly difficult as it can be both pliable and rigid simultaneously. We had to be cautious with each modification made to the box so as to not render it useless.

Another issue that we encountered was working separately on the circuit designs of the sensors. We did most of our communication via Discord Voice Calls as we worked on the sensors indirectly together. Before we began the initial process to create a smart safe system we had to have the same sensors. Both of us had to buy a separate kit of sensors. We were unable to purchase the exact same kit (out of stock on Amazon at the time) which resulted in one of us having the wrong photoresistor sensor. The one pictured in *Figure 2* was the one used by Chip. The one Samip had was slightly different in that this sensor was missing an extra analog output pin as well as a blue screw mounted on the sensor to control the sensitivity.

Teammate Responsibilities:

Both Samip and Chip shared the same responsibility of getting the sensors to work on both of our Raspberry Pis. We wanted to mirror each other's work due to us working remotely, so we both made sure to get sensors working so we both understood how everything worked. In addition to the sensors, Chip also wrote the Python code for the safe and put everything into the safe container itself. These steps were a bit more difficult to share remotely, so Chip took the initiative to complete them himself while consulting and discussing with Samip.

Conclusion:

The original smart safe idea came from Chip when he had realized that he needed to get into his own personal lock & key safe. He had lost access to the safe since he had misplaced his keys to unlock it. Therefore, the intention behind a smart safe came about. Then he pitched the idea to Samip after realizing that the Google Assistant idea was a bust. The design of the safe came about fairly quickly once we realized that we needed to come up with a makeshift design. The safe consists of five different sensors; servo motor, motion sensor, LCD display, keypad, and a photoresistor module. Together these five sensors make up the core design of the smart safe and are a critical part of the system. Housed inside the makeshift box with a cut to open and close a safe. This was our project idea and design as there are many more complex smart safe systems compared to our own. Through this, we were both able to get a better understanding of sensors/components used in real-life applications.

YouTube Link:

https://youtu.be/s7c2CdLf1TA

Works Cited

21, NEIL IVES March, et al. “HC-SR501 Pir Motion Sensor Arduino Tutorial (3 Examples).” Makerguides.com, 28 Sept. 2021, www.makerguides.com/hc-sr501-arduino-tutorial/.

262588213843476. “Python Script for Controlling a Servo Motor (Tower Pro SG90) Using Raspberry Pi.” Gist, gist.github.com/elktros/384443b57a33f399a4acba76191e0e63.

“4x4 Matrix Membrane Keypad Switch Raspberry Pi Arduino 5056240315454.” *EBay*, https://www.ebay.com/itm/284430889670.

Engr Fahad. “Raspberry Pi 16x2 LCD I2C Interfacing and Python Programming.” Electronic Clinic, 25 Nov. 2021, www.electroniclinic.com/raspberry-pi-16x2-lcd-i2c-interfacing-and-python-programming/.

Frederic, and Name *. “HC-SR501 PIR Motion Sensor on a Raspberry Pi.” Freva.com, 16 Aug. 2021, www.freva.com/hc-sr501-pir-motion-sensor-on-raspberry-pi/.

“How to Setup an I2C LCD on the Raspberry Pi.” Circuit Basics, 14 Nov. 2021, www.circuitbasics.com/raspberry-pi-i2c-lcd-set-up-and-programming/.

John Howard 53.3k2121 gold badges4646 silver badges6464 bronze badges, et al. “Run Certain Code Every n Seconds.” Stack Overflow, 1 Sept. 1958, stackoverflow.com/questions/3393612/run-certain-code-every-n-seconds.

Last Minute Engineers. “How HC-SR04 Ultrasonic Sensor Works &amp; How to Interface It with Arduino.” Last Minute Engineers, Last Minute Engineers, 18 Dec. 2020, lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/.

peppe8o. “SG90 Servo Motor with Raspberry Pi Pico and MicroPython.” peppe8o, 3 June 2021, peppe8o.com/sg90-servo-motor-with-raspberry-pi-pico-and-micropython/.

“Photoresistor LDR Module.” Twins Chip, https://www.twinschip.com/Photoresistor-LDR-Module.

Staff, Maker.io. “How to Connect a Keypad to a Raspberry Pi.” DigiKey Electronics - Electronic Components Distributor, Maker.io, 19 May 2021, www.digikey.com/en/maker/blogs/2021/how-to-connect-a-keypad-to-a-raspberry-pi.

SunFounder IIC I2C TWI 1602 Serial LCD Module Display. www.amazon.com/SunFounder-Serial-Module-Display-Arduino/dp/B019K5X53O?th=1.

TowerPro SG90 Micro Servo Motor. vayuyaan.com/product/vayuyaan-towerpro-sg90-micro-servo-motor-180-degree-rotation-blue/.

“Using Light Sensor Module with Raspberry Pi.” UUGear, www.uugear.com/portfolio/using-light-sensor-module-with-raspberry-pi/.