Facility Related Control System

May 2, 2024

Wesley Cooke, Brycen Havens, Lauren Murach, Chris Stickney

A computer science senior capstone project for Augusta University with funding from Trideum Corporation.

AUGUSTA UNIVERSITY

TRIDEUM®

Contents

1 Introduction

1.1 General Description

The Cyber-Physical Teaching Kit (CPTK) is a miniature, modular building with a fully-functional control system. The belief is that by building a robust model, students will better understand how to think about potential cyber attacks. The code is mostly written in python for the Raspberry Pi 4, along with C++ code for the Arduino. Students are more likely to encounter python and Arduino coding throughout high school or college, which is why these two languages were chosen.

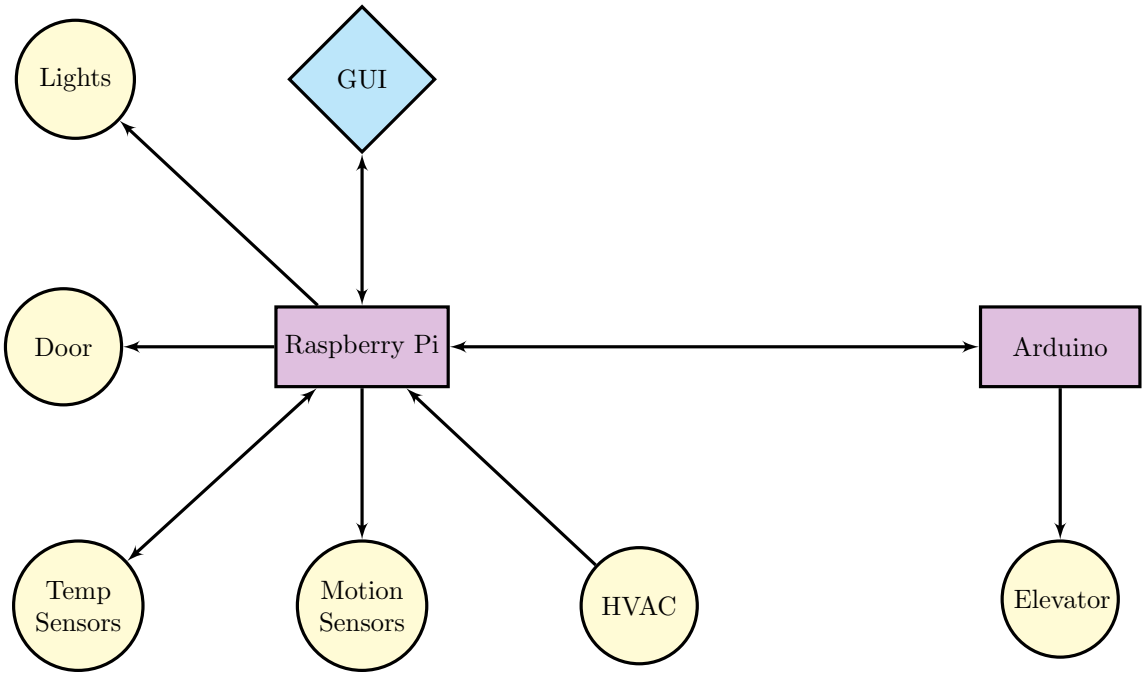
1.2 Features

- Sturdy, lightweight 3D printed frame with files available for re-printing
- Accurate elevator with button controls on each floor and a re-calibration function
- Temperature monitoring with an HVAC and duct work system for air conditioning
- Security door system using RFID badges with a working door
- Lights with individual floor motion detection
- Full database for logging events, errors, and storing user configurations
- Security assessment in pdf form detailing vulnerabilities, mitigation, and impact

1.3 Links

- Github for open source code under the MIT license
- Security Assessment PDF

1.4 Block Diagram



2 Getting Started

An already constructed version of the CPTK was built for Trideum Corporation. The following are instructions for running the program on the existing system. If a part of the kit breaks, or a second copy of the kit is desired, see page 4.

2.1 Running the App with a Monitor

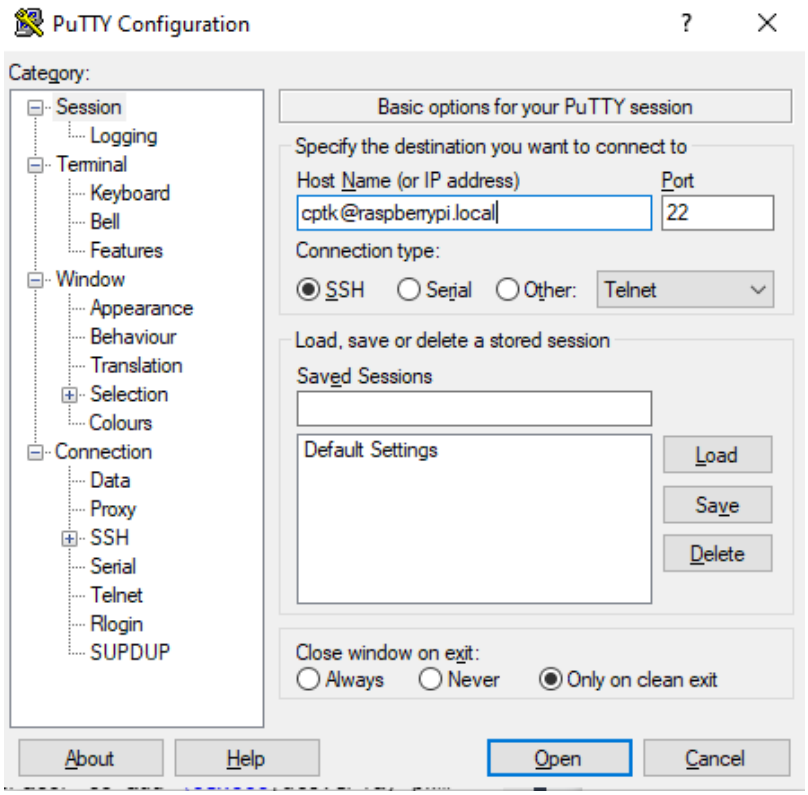
This is the most ideal way to view the application, as the GUI is visible and intractable. If a monitor is not available, see the next section.

1. Plug an HDMI cable, keyboard, and mouse into the Raspberry Pi and power the system. The Pi will boot automatically.
2. Open the console using the icon at the top right of the screen or with the keyboard shortcut `CTRL+ALT+T`
3. Run the application with the following command:
   `sudo python3 AU_Capstone_Trideum/application.py`

2.2 Running the App without a Monitor

Note that the GUI will not be accessible with this method.

1. Plug an Ethernet cable into the Pi and power the system. The Pi will boot automatically. Make sure your Ethernet settings allow internet and file sharing.
2. For Windows users, install PuTTy.
3. Run the .exe file and enter in the following for the host name, then press enter:
   `cptk@raspberrypi.local`



4. For macOS/Linux users, open the command prompt and enter the following command:
   `ssh cptk@[raspberrypi_ip_address]`
   To get the IP address, run ipconfig in the command line. Alternatively, follow the instructions for Windows and install PuTTy.

5. Once connected, the command line will prompt for a password. The password for the cptk user is `CPTK-Project`
   The text will not show up while typing the password, but hitting enter will connect to the Pi.

6. Run the application with the following command:
   `sudo python3 AU_Capstone_Trideum/application.py -platform offscreen`

3 Kit Repair or Duplication

If a piece of the kit disconnects or breaks, the following sections detail how to order, manufacture, and connect replacement parts.

3.1 Frame Construction

The following folder on Github contains the files of all printable files. After the name, in parenthesis, contains the quantity of each of the files required to print. All screws and their quantities are in the section 3.2. For best results, a soldering iron is required to perform plastic welding.

3.2 Part Ordering

The following are the parts used in our final design and possible locations of where to buy them. Note: the quantities used are how many individual pieces are used for our design and not how many packages to order. In almost all scenarios, only one pack needs to be ordered.

Part Purpose	**Quantity**	**Part Number/Link**
Basement Temperature Sensor	1	TC74 A0
First Floor Temperature Sensor	1	TC74 A1
Second Floor Temperature Sensor	1	TC74 A5
Motion Sensor	3	SR312
Servomotor	4	SG90
5mm LED	2	Through-hole 5mm LEDs
Light Strand	1	Neopixel Light Strip 5V
3 Pin Connectors	4	BTF-LIGHTING JST SM
RFID Scanner	1	HiLetgo - Mifare RC522
HVAC Cooling System	1	Thermoelectric Peltier Cooling System
Stepper Motor Driver	1	Stepper Motor Driver TB6600
5V Buck Converter	2	12V Step Down to 5V Buck Converter
Power Block	1	12V 20A 240W AC/DC Power Adapter
Frame Screws	50	10*1-1/4-50

3.3 Part Wiring

Every part is connected to a strand of wires that run to the Pi/Arduino. Each of these is individually labeled with a letter. The individual wire column is if the wire is held so that the letter is facing towards you, and the wires are read from left to right. Note that many times colors will seem unintuitive (such as using red for ground or black for power). This is due to the limitations of the ribbon cables used for our project. For future versions, wire crimping with connectors, such as Molex connectors, should be used.

Part Lettering	Part Purpose	Individual Wires (Left to Right)
Raspberry Pi Parts		
A	Basement Temperature Sensor	GPIO 2, GND, GPIO 3, 3.3V
B	First Floor Temperature Sensor	
C	Second Floor Temperature Sensor	
D	Basement Motion Sensor	GND, GPIO 12, 5V
E	First Floor Motion Sensor	GND, GPIO 16, 5V
F	Second Floor Motion Sensor	GND, GPIO 20, 5V
G	Basement HVAC Vent Servomotor	GND, GPIO 19, 5V
H	First Floor HVAC Vent Servomotor	
I	Second Floor HVAC Vent Servomotor	
J	Neopixel Light Strand	GND, GPIO 21, 5V
K	HVAC Enable Pin	GPIO 6
L	RFID Green LED Indicator	5V, GND
M	RFID Red LED Indicator	
N	Door Reed Switch	GPIO 22 GND
O	Door Servomotor	GND, GPIO 18, 5V
P	RFID Scanner	GPIO 8, GPIO 11, GPIO 10, GPIO 9, GND, GPIO 25, 3.3V
Arduino Minima Parts		
Q	Bottom Micro Switch	5v, Digital Pin 3
R	Upper Micro Switch	5v, Digital Pin 2

3.4 Raspberry Pi Configuration

The following are instructions on how to configure a new Raspberry Pi 4 Model B with all required dependencies. You will need a 32gB micro SD card and a micro SD card reader on another computer.

1. Install the 64-bit Legacy Raspberry Pi 4 OS from the Raspberry Pi website and extract the downloaded file file.
2. Many times the official OS-imager does not correctly flash to the SD card, so install Balena Etcher.
3. Open Balena Etcher, upload the extracted Raspberry Pi OS file, and click "Flash."
4. When the image is complete, eject the micro SD card and slot it into the Raspberry Pi.
5. Using the Pi without a monitor is possible, although not recommended. To do so, plug the micro SD card back into your PC, navigate to the boot folder, and add two files.
   (a) A blank file called `ssh` with no extension
   (b) A file called `userconf` with no extension with the copied contents below (the text is intentionally tiny to fit on the page, it contains the password CPTK-Project, but encrypted.
   `cptk:$6$DovL/2E2U/6Eu0zk$Ju7iUqpg5hMMS8Dn5VlCrWACggQlGCYJEviPv0apOHPpD9bxVAXwv9Uvaykhm0WJcjdPHvz2.xVvgIwJtx29U0`
6. Plug in the Pi and follow the instructions on the monitor. Alternatively, boot up the Pi and ssh into the Pi from another computer. See section 2.2 for more details.
7. Download the required repositories by running the following
   ```
   sudo pip3 install smbus2
   sudo pip3 install rpi-hardware-pwm
   sudo pip3 install pi-rc522
   sudo pip3 install adafruit-circuitpython-neopixel
   ```
8. Edit the boot/config.txt file as an administrator to add the text `dtoverlay=pwm-2chan` at the bottom of the file. This file can be opened as an administrator with the following command in the terminal:
   `sudo nano /boot/config.txt`

9. Note: This step is ONLY if you are an Augusta University Project Team working on continuing our project! If the AU-Guest network does not connect to the internet, go to www.augusta.edu on the Raspberry Pi, and hit accept on the network. Afterwards, reboot the Pi.

3.5 Cloning the Code Repository

If the Pi is connected to the internet, then run the following to copy the contents of the required code onto the Pi in its own folder:

```
git clone https://github.com/lmurach/AU_Capstone_Trideum.git
```

If it is not connected to the internet, then run the same command on your PC and transfer the files with scp or a flashdrive.

3.6 Setting up the Arduino Minima

The Arduino Minima has its own code folder than should either be cloned onto your desktop or downloaded independently from our Github. It should be in its own folder. The steps for flashing the appropriate code onto the Arduino is the following:

1. Download the Arduino Legacy IDE from the official Arduino Website. At the time of writing this, the Arduino Legacy IDE should be used as the new Arduino IDE is too new to guarantee software stability. If the Arduino Legacy IDE is deprecated, than the newest version should be used.
2. Download or clone the Elevator folder from our Github.
3. Plug the Arduino in through USB. It will power automatically.
4. Open the elevator.ino file into the Arduino Legacy IDE.
5. Open Tools>Board>Board Manager. Search for Minima and install the R4 Minima Board Package.
6. Select the R4 Minima board and the port currently connected to the Arduino. The IDE will detect the Arduino and give the correct option. If no ports are available, restart the Arduino IDE.
7. Press the upload button and wait for the console to show that the code is finished uploading. The code will remain on the Arduino even when power is off.
8. After successfully uploading the code, connect the arduino minima to the raspberry pi.

4 Detailed component description

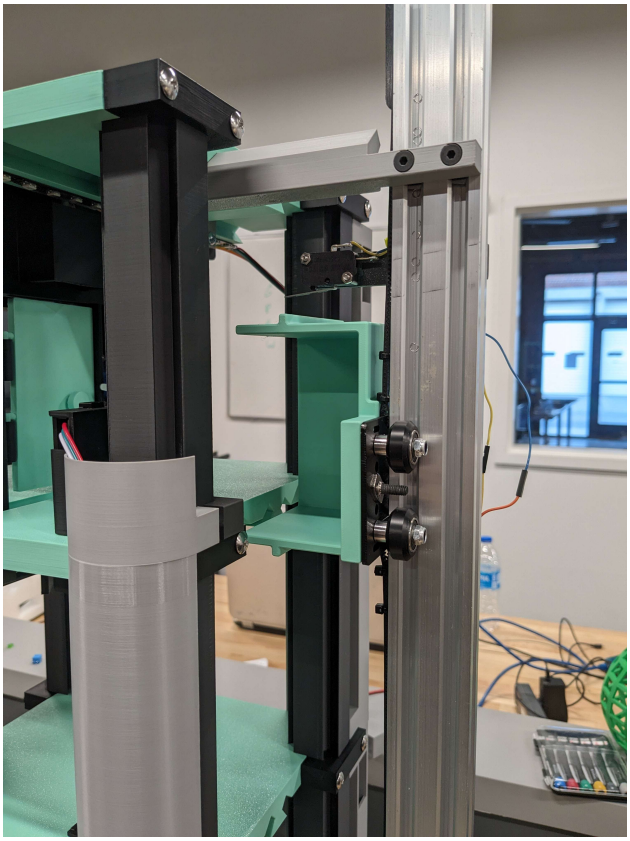
Below the general overview of code, design, and hardware decisions are explained. In cases where the circuit diagram is not a wired connection directly from the Pi to the component, a circuit reference is given.

4.1 Frame

The design of the frame is a modular 3D print with sturdy filament. This design allows parts of the frame to be replaced. The specifications of the printer the initial CPTK is printed from is the Bambu Lab X1-Carbon with Bambu filament. However, the design is sturdy enough to allow for most printer and filament types.

The side posts are designed around extruded aluminium cutouts, but with the extra design flexibility with the sensors mounting locations. A floor for each level has similar cutouts to mount the door, elevator, HVAC cooler, and temperature sensor within the middle of the frame. These pieces are easily removable by sliding them in and out, which gives the opportunity to demo physical vulnerabilities or attacks on the sensor busses.

4.2 Elevator



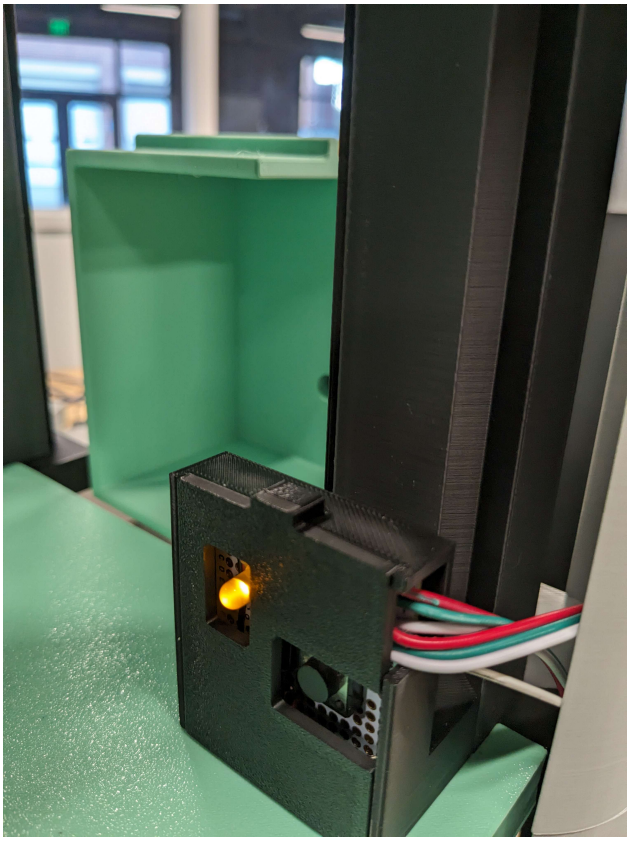
The elevator system has a mounted button panel on each floor where a user can push a button to request the elevator. When the elevator reaches a floor, it briefly stops to allow for passengers to depart. Both floors not currently being serviced can request the elevator, and the elevator will visit the floors on a first-come, first-served basis (FIFO Queue). Pressing a button multiple times will not cause an elevator to remain on a floor or interrupt the queue in any way.

The elevator is the only component not included on the Raspberry Pi. This is due to two main reasons:

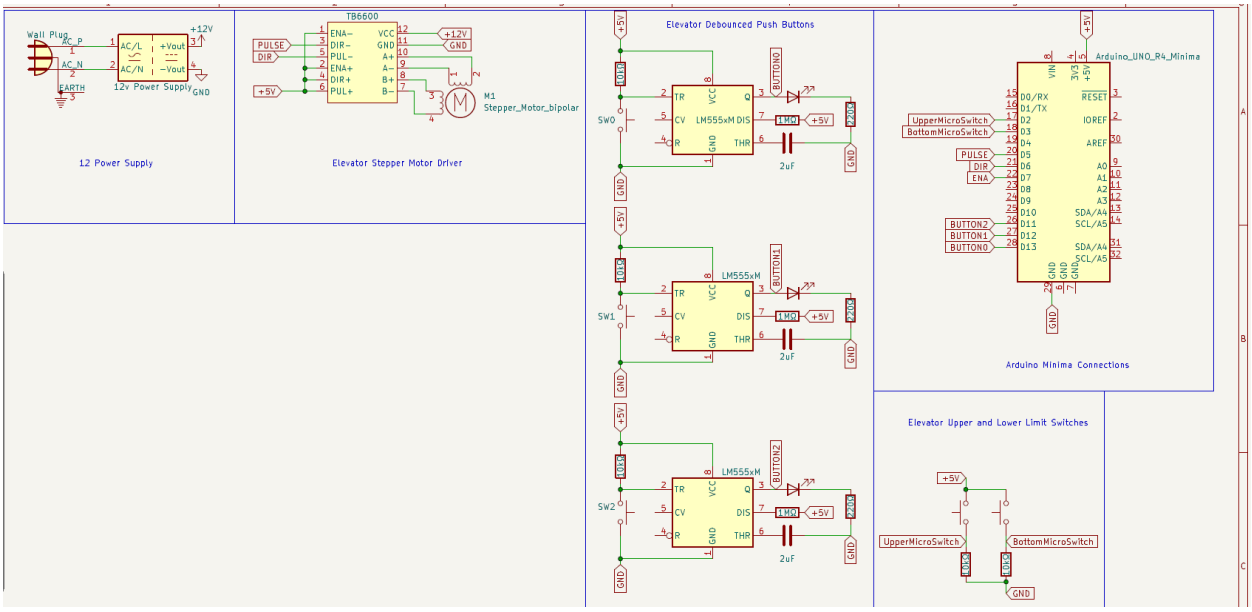
1. The Pi only has 28 general purpose I/O pins, which is too few for the amount that the CPTK requires to operate. The elevator is the most demanding on this pin count, initially requiring 13 GPIO pins (this later was changed to 8).

2. The elevator must poll the buttons on each floor and appropriatley respond to them being pressed in a timely manner. The program on the raspberry pi could not handle the constant polling. Additionally, the stepper motor makes use of delays which would be blocking calls on the raspberry pi. Moving the elevator system to an Arduino eliminated these issues.

The elevator is on an Arduino R4 Minima and communicates to the main Raspberry Pi over USB (UART). This connection consumes only a USB port on the Pi and requires much less strict timing requirements. The Arduino sends and elevator state update no faster than every 500 milliseconds. This state update includes what floors are currently in the requested queue and which floor the elevator is currently at. The raspberry pi appropriately displays this information on the GUI.



The button on each floor utilize a 555 timer chip to help de-bounce the button signal and give the Arduino a chance to "see" the signal in case the elevator is currently moving. The 555 timer chip will keep one button press active for about 2 seconds. All circuitry for the Arduino/elevator can be found here.

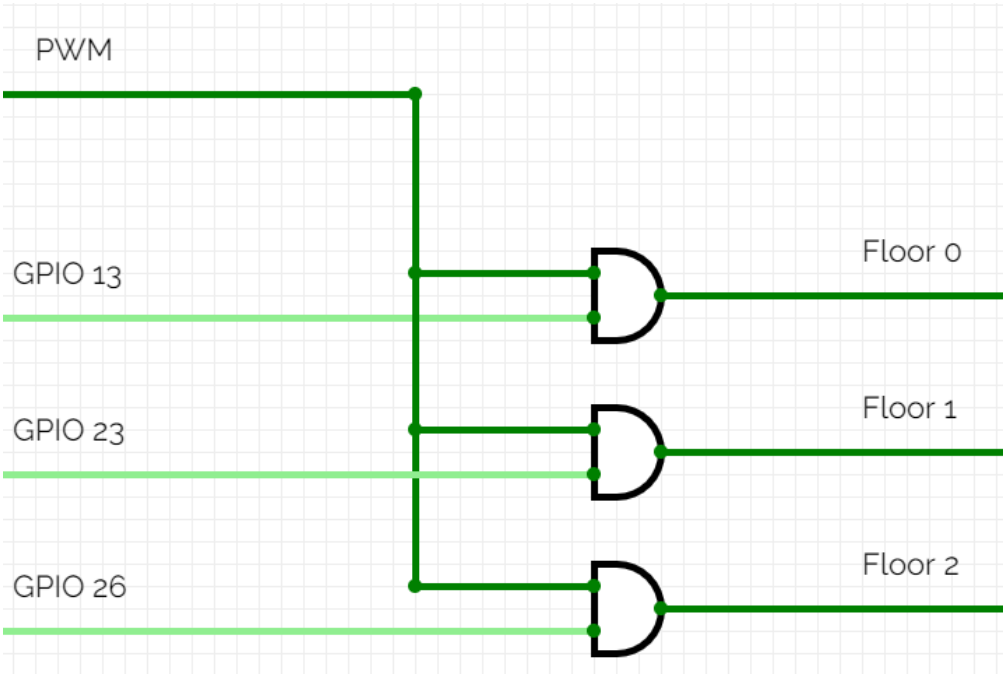


4.3 HVAC

The HVAC system contains a cooler, controlled by a single pin, and a vent on each floor, controlled by a servomotor. When the any of the floors require cooling, the cooler turns on. If any of the individual

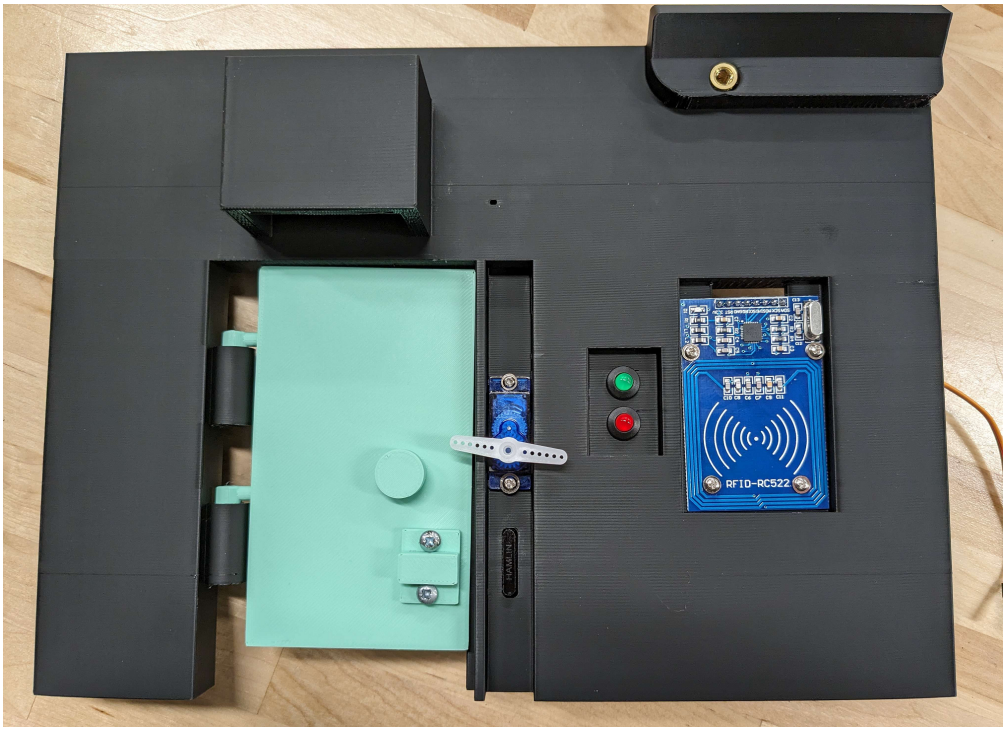
floors require cooling, the vent for the required floor opens. To avoid the vents from constantly opening and closing, software hysteresis is programmed into the temperature. A temperature must drop below the set temperature two degrees before turning off or raise above the set temperature two degrees before turning on.

The servomotors are controlled by a single PWM pin and some AND gates. Hardware PWM MUST be used as the Linux operating system is not consistent enough to guarantee the timings required. The motor will constantly wiggle from inconsistent timings if software PWM is attempted. The AND gates are utilized due to the fact that the Raspberry Pi only has two PWM pins. Another possible implementation is using a de-multiplexer. This solution did not work for our group due to the signal "leaking" occasionally and causing the servomotors to open when it is not their turn. See below for the circuit used.



4.4 Security Door System

The door is a slide-in attachment to the top floor, facing the elevator. Its purpose is to emulate a door preventing access to valuable assets, such as servers. In order for an employee to enter, they must have a valid ID card. If the card is valid, the green light turns on and the door is unlocked. The lock remains unlocked until the door is closed. If an invalid ID card is scanned, a red light turns on instead.



The RFID works by frequently checking if a card is in proximity by attempting to read a valid card UID. Once a card is in range, it reads the card's UID and data bytes. This information is checked against the current employee database, stored in a sqlite file. Three conditions must be met for the door to open from a card scan.

1. The user's data must match an employee in the database.
2. The employee in the database must have door access privileges.
3. The UID on the card must match the UID stored in the employee's database row. This is to help mitigate card tampering.

Alternatively, the door lock changes state from the GUI, with the lock and unlock buttons. The door is not able to lock if the door is open, to prevent the servomotor from being broken.

When the door is unlocked, the door is able to be opened by pulling on the handle. Inside the door is a magnet. Next to the switch (below the lock) is a reed switch. The magnet closes the switch when far away and open the switch when it is close. This interaction allows for the Pi to know if the door is open or close by using an active-high pullup resistor.

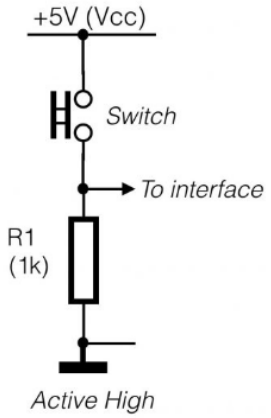


Image from www.ipson.nl

4.5 Motion Detectors

The motion detectors are mounted to their own 3D printed mounts, which fit in the frame through the side to the center of the frame. Their purpose is to detect motion on their respective floors and to send signals to the controlling Pi when motion is sensed. The Pi will then activate the lights on that floor, with the color depending on whether or not the building is open. If the building is open, the lights turn white. If the building is not open, the lights turn red and an after-hours motion alert is written to the logs. The lights will turn off if no motion is detected for a certain time after the lights are turned on, emulating a real motion activated lighting system.

The motion sensors work by detecting changes in thermal signatures in the surrounding area. Large enough changes in a short time frame indicate motion. The sensitivity of the sensors may be adjusted in the code by changing the threshold argument value in the declaration of each motion sensor. When motion is detected, the GUI will update and show that motion has been detected from that sensor on it's floor.

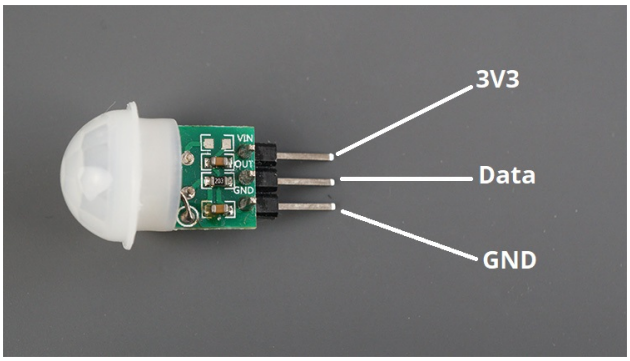


Image from www.randomnerdtutorials.com

The image above depicts the pin out of the motion sensors we used. The middle data line must be connected to a GPIO pin associated to a floor to send readings, and the sensor must also have a power and ground to function. (See section 3.3 for specific pins in our application.) When the data line sends a high signal, the Pi recognizes that there is motion on that floor. If there is a low signal, the Pi interprets that as there being no motion.

4.6 Neopixel Lighting System

The lighting system is implemented using Adafruit Neopixels. Neopixels require two special libraries to properly function on the Raspberry Pi. These libraries are the Adafruit Blinka library and the Adafruit Blinka Neopixel library. Information on these libraries is readily available online. The Neopixels are able to be indexed with these libraries, allowing control over specific ranges of the Neopixel's lights.

During normal system operation, the lights should be in an off state. If motion has been detected within the past 5 seconds and the alarm is disarmed, the lighting should turn white. If motion is detected while the alarm is enabled, the lights will instead turn red for a period of 5 seconds. After a delay, the lights shut off, emulating real implementations of motion activated lighting.

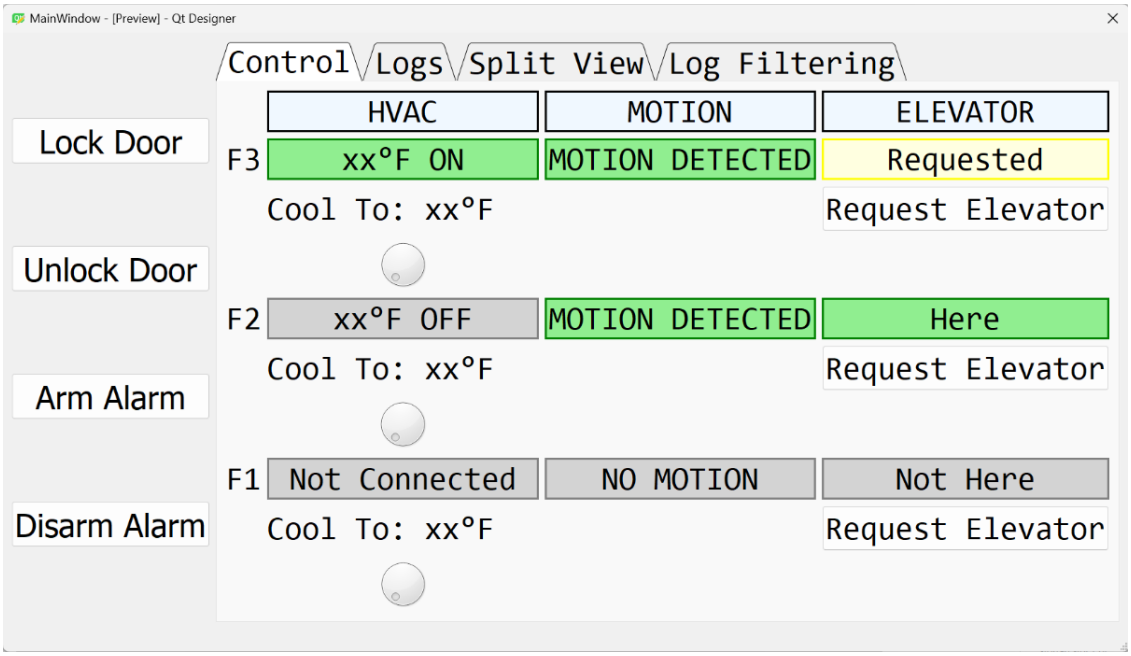
4.7 Graphical User Interface

The CPTK system incorporates a GUI to provide additional control and monitoring over the building sensors. The main screen provides the ability to view the current temperature, whether the HVAC is on, whether motion is detected, and the elevator status on every floor. In addition, there are buttons to lock/unlock the door on the second floor and arm/disarm the building.

Currently the following functionality is implemented:

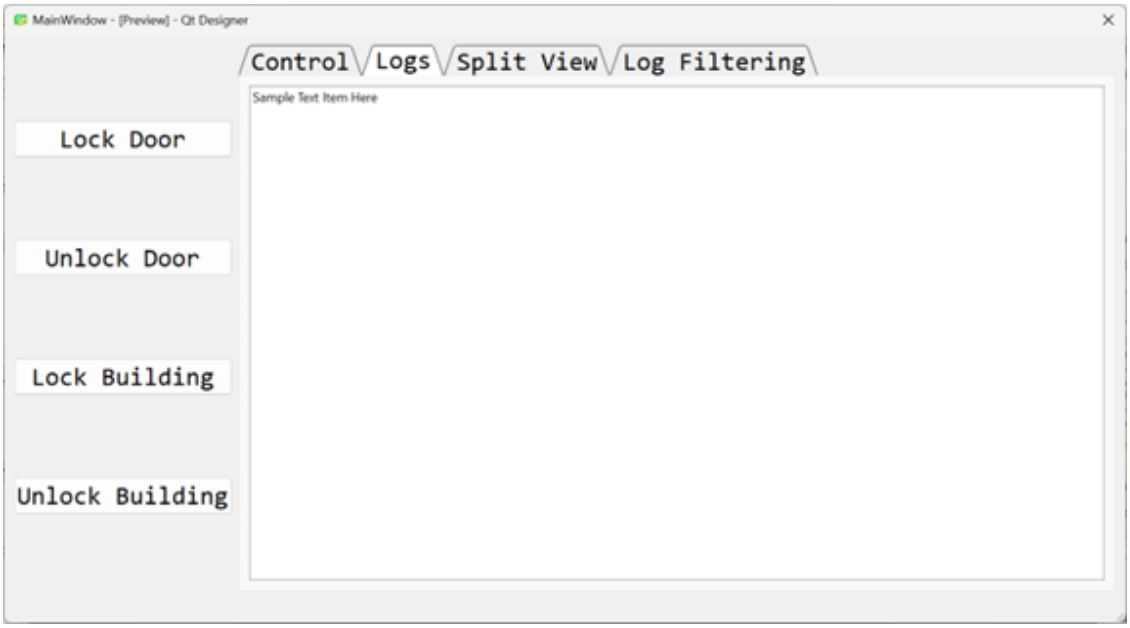
- Unlock and Lock door buttons work with physical system.

- Dial changes the "cool to" temperature.
- Buttons generate appropriate logs in the database.
- Logs can be filtered according to some criteria.
- Temperature labels show the current temperature.
- Motion blocks light up when motion has been detected.
- Arduino communicates the state of the elevator to the GUI.



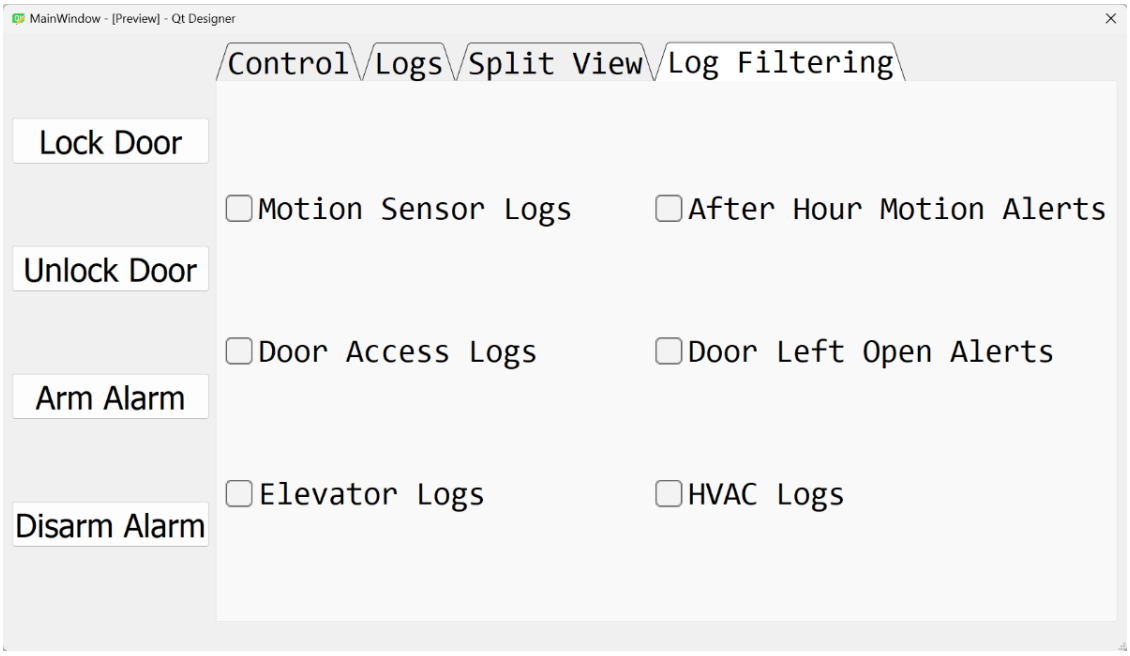
Control View:

The control view shows the overview of all the systems. Notice the floor labels on the left column: F1, F2, and B which follow typical elevator notation. Units are displayed in degrees Fahrenheit, and the dial has tick marks to help it stand out. Green coloring indicates that a certain system is on. Grey coloring indicates that it is off. Red coloring indicates an error.



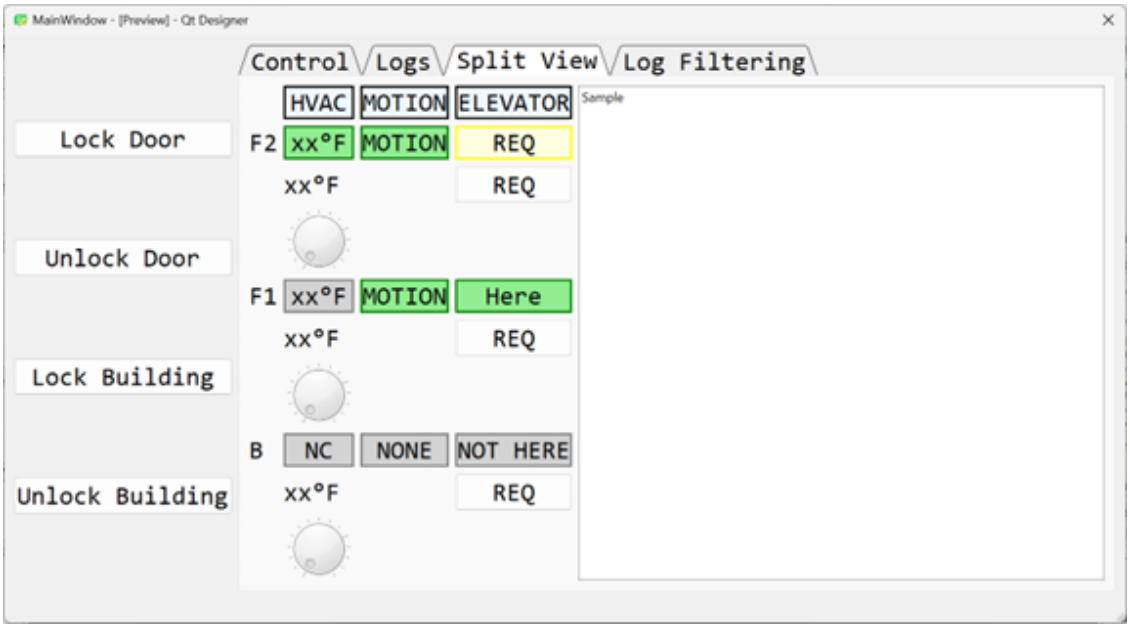
Log View

Displays the logs and filtered by date so that the most recent log is at the top.



Log Filtering View:

Each sensor in the building produces database logs when changes occur. Component errors (when some sensors are disconnected) also log errors to the database. The logs list the time of the error and the error type in a readable format. Since so many logs are possible, logs are filterable by type in the Log Filtering tab of the GUI. As many or as few log types can be selected in this panel.



Split View:

The split view is a combination of both the control and logs view. Notice that some information from the control view is lost to adapt for the space occupied by the logs.

4.8 Database

The database is a sqlite database, a relational database that stores data locally in a .sql file. This format allows for the ability to easily query for different logs and sort by date. CPTK's database also has a function to filter individual queries based on user input. It also converts from a database row to an easy-to-understand sentence format for the GUI.

A configuration table stores the last set values from the GUI into a table so that important settings such as temperatures do not need to be reset. This table is updated upon closure of the program.

The following is the ER diagram of the database: