GIX MSTI Program

TECHIN 512, SENSORS and CIRCUITS

Final Project

Bed Occupancy Detection using Force Sensitive Resistors (FSR)

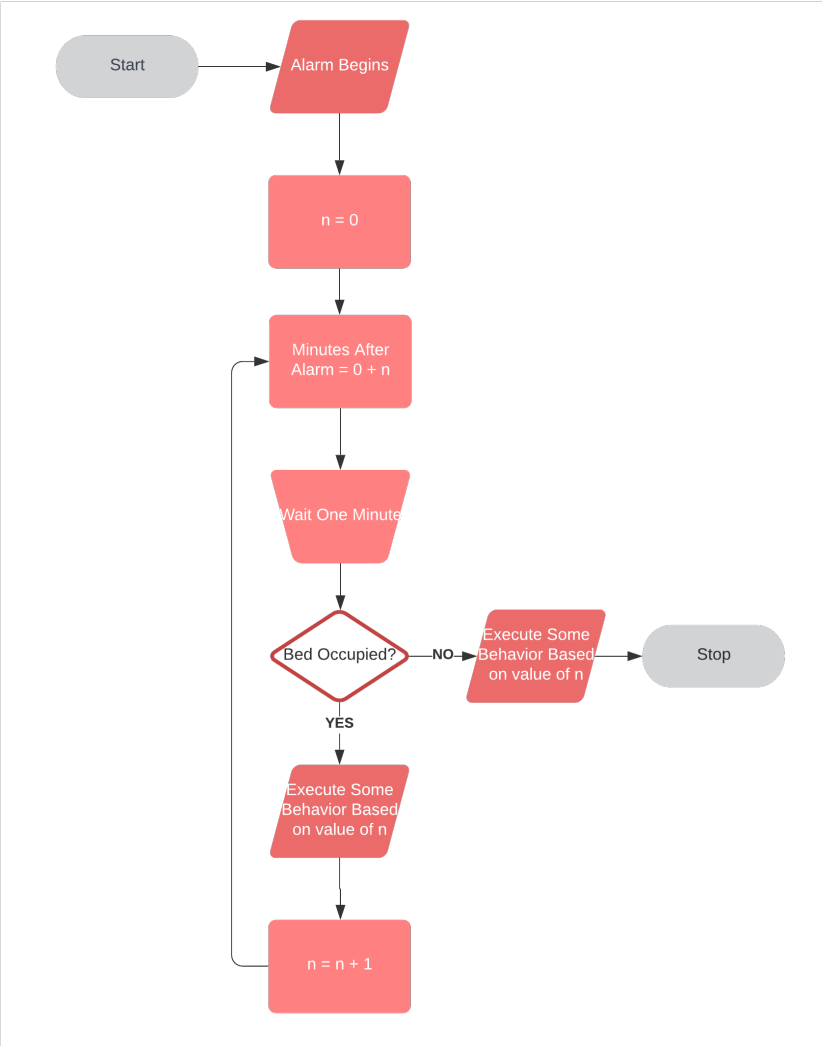
3-10-20

Cody Gagnon

Introduction

Project Goal

When one begins building out IoT devices for their smart home, they begin to want to automate everything; or more specifically, to automate all the *things*. While browsing the Internet I came across a Reddit post asking what user's favorite "in bed" presence detection systems are. I became fascinated by the idea that one could create automations based off of the state of whether their bed is occupied or not. The goal of this project is to take build a system which helps users save time in the mornings by encouraging them to get out of bed when their alarm goes off. I like to think about it as "automating getting out of the bed." The beauty of this system is that since it is meant to be integrated into a home automation platform, much of the work and possibilities for its use come out of software written for the platform (with systems relying on feedback from the hardware that detects bed occupancy). The basic idea for my implementation of a wake-up system is as follows:



Flowchart of Alarm System

Based on the value of *n*, various levels of annoyance will increase until the user is out of their bed. If the user gets up within a minute of their alarm starting, there are no consequences. Using this negative reinforcement will help curb the user's willingness to sleep in past their alarm time. In addition, positive reinforcement can be added depending on the value of n as well.

Use Cases

There are also several use cases that arise when performing bed occupancy detection in regard to a home automation system. Some of the automations I've setup or am planning to setup with this sensor system include:

- If both my partner and I are in bed and one of us leave, turn on a night light
- If both of us are in bed then change the state of all affected devices to a mode suitable for sleep (TV off, lights off, alarm system armed, etc.)
- If someone in the home is in their bed for an extended period of time, send a notification to check on them

And other useful automations. I discovered that it is possible to use two of these sensors to tell the difference between whether it's my partner or I sleeping. What I found is that I move a lot more in bed than she does, and thus we can use this to discern which side of the bed either of us is on. But it could also be used to determine if a guest is sleeping in our bed and ensure that "guest mode" for our home automation system is activated. This extra step of recognition is pretty cool and something I didn't immediately understand is possible with this sensor system. You can read more about that use case and my findings in a paper I wrote for another class on it (must be logged into UW; if you want to read it but aren't a UW student please email me at hello@cody.codes): http://bit.ly/2INUdud.da

User Profile

The typical user profile is currently a home automation enthusiast, though it is by no means restricted to this profile. The hardware and software setup of this system is easy enough for anyone to order the parts from Amazon and set it up on their bed within the span of a day. In the future I believe this sensor or something with equivalent functionality will become even easier to implement. Especially because this is an unobtrusive solution which requires zero devices to be attached to the user and can be safely powered underneath a bed, it is an appealing product for users of many demographics. Past research with these types of sensor systems have been done in several environments, but most notably, nursing homes would benefit for some of the useful automations I've outlined above.

Sensing Requirements

Based on the goal and use cases, we require a sensor which is able to distinguish the state of a bed with and without a user while staying under the user's bed. This means that the weight of the bed will also affect the sensor and should be accounted for. The most important

characteristic of this sensor is that it is simply able to distinguish when a human's body weight + the weight of the portion of the sensor which is under the bed vs. when the sensor is under the bed with no human on it. We can thus use the following physics equation:

$$W = \left(mass\ of\ a\ person\ (kg) + mass\ of\ a\ bed\ (kg)\right) * 9.8\ (m/s^2)$$

Our sensor system is aimed for the mass of an average human being and the mass of an average bed. There will be some tolerance in regard to the mass of a person and the mass of a bed, but as long as the sensor is not maxed out by the mass of the bed, we should be able to distinguish the person being on/off the bed. What I've found in my research is that the two different types of sensors which would be applicable for being under a bed each have a different tolerance:

1. An FSR, or Force Sensitive Resistor, has a maximum of 10kg which can be applied on a single spot or anywhere on a strip
2. An array of load cells can sense their weight in a specific location. They should be placed on the feet of a bed and should be rated at 25kg to ensure they can handle the weight of the bed and bed frame.

One important note is that a waterbed probably won't work for either of these sensors as it has a weight too high for a single point and as a whole, and thus will max out the value for each of these sensors before a user gets onto a bed, making it indistinguishable. For this type of bed, a different sensor system entirely, using something like ultrasonic at the foot of the bed would suffice.

Sensor Selection

When comparing sensors, I wanted to ensure that the user experience would be forefront; if cost would become higher but initial setup is reduced, so be it. I want to create an easy sensor which "just works" after it's setup. All of these sensors are meant to be placed on or under a bed; any solution which isn't sensing contact with the bed wasn't considered. To reduce initial setup time, I also wanted to purchase a sensor package rather than creating my own custom hardware sensor. That being said, I came across a fascinating article explaining that it's possible to create a capacitor with very low cost which is able to detect when a child is in a car seat (which is similar in principle to this project). You can find that work here and I've also included it in the table below.

	FSR (2)	**Load Cell (4)**	**Vibration (4)**	**Capacitive (2)**
Price	$30.60	$7.80	$1.72	$0.78
Setup	Easier	Harder	Harder	Easier
Calibration	No	Yes	Yes	N/A
Long-term Use	Yes	No	No	N/A
Complexity	Less	More	More	More

Tabular Comparison of Sensors

From the above table, while we can see that while the FSR, or Force Sensitive Resistor, has the highest price, it also comes with easy setup involving no calibration, overall being less complex and suitable for long-term use (the Amazon product page states it's durable for use > 1 million

times [but the cheapest price is from the AliExpress link in the table]). Other sensors, such as the load cell and vibration sensors, require significantly more overhead as they need to be calibrated and require setup for the four posts of a bed. While they may be able to provide more information, such as weight, this isn't a concern for our sensor system as we only want to detect bed occupancy. In fact, the vibration sensors may not work depending on the context (I live next to the Interstate where vibrations from trucks and other vehicles may interfere with bed occupancy detection). Long-term these sensors may need further calibration and could give out depending on how they are handled. The capacitive sensor is unknown as it's a custom sensor which may or may not be able to withstand long-term use and may or may not need initial calibration. The load cell, vibration, and capacitive sensors are all more complex as they require a better understanding of physical phenomena related to the context they're used in to accomplish the sensing goal of bed occupancy detection. For the ease of installation, reduction of overall complexity, not needing to initially calibrate the sensor, and known long-term usage, the Force Sensitive Resistor is the best for the project!

Prototype Design

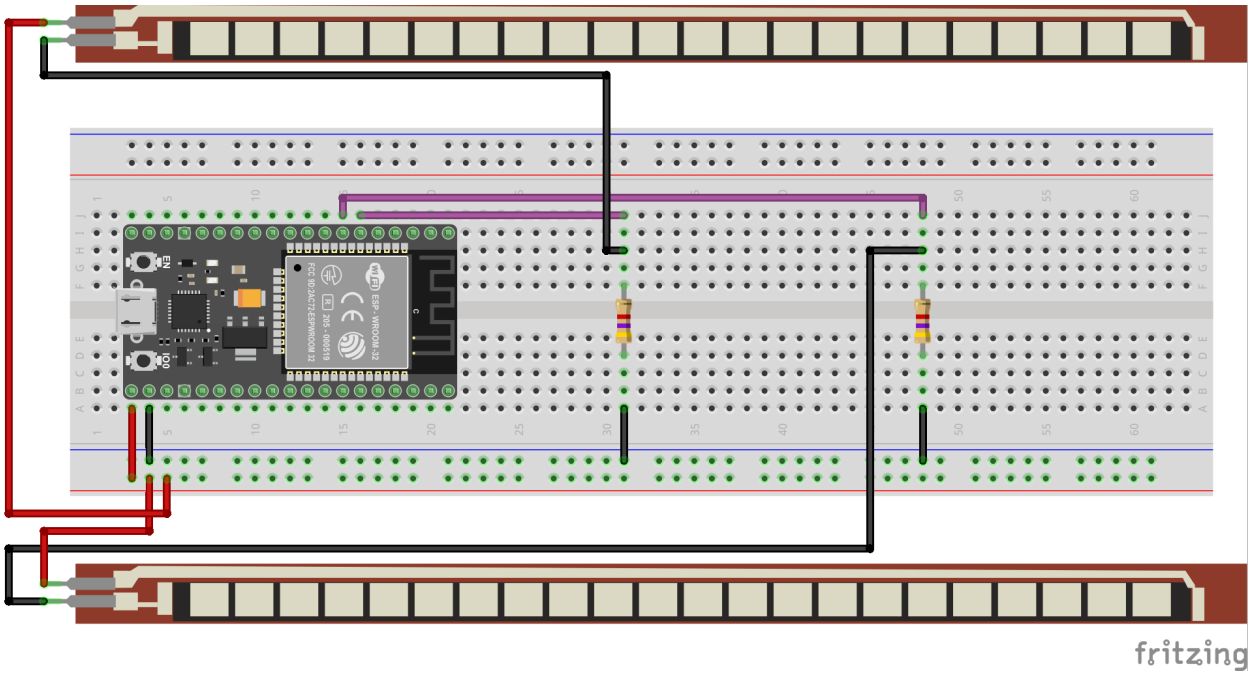
Hardware

I am using an ESP32 and its two onboard ADCs to detect values from an FSR, or Force Sensitive Resistor. It is powered via a standard USB power supply with 5 V. These are sensors which decrease in resistance (Ω) as pressure is applied to them. The particular FSR I'm using is the SF15-600, an approximately 24", rectangular FSR, which is rated for up to 10kg.

Walfront SF15-600
amzn.to/3aoAkWs

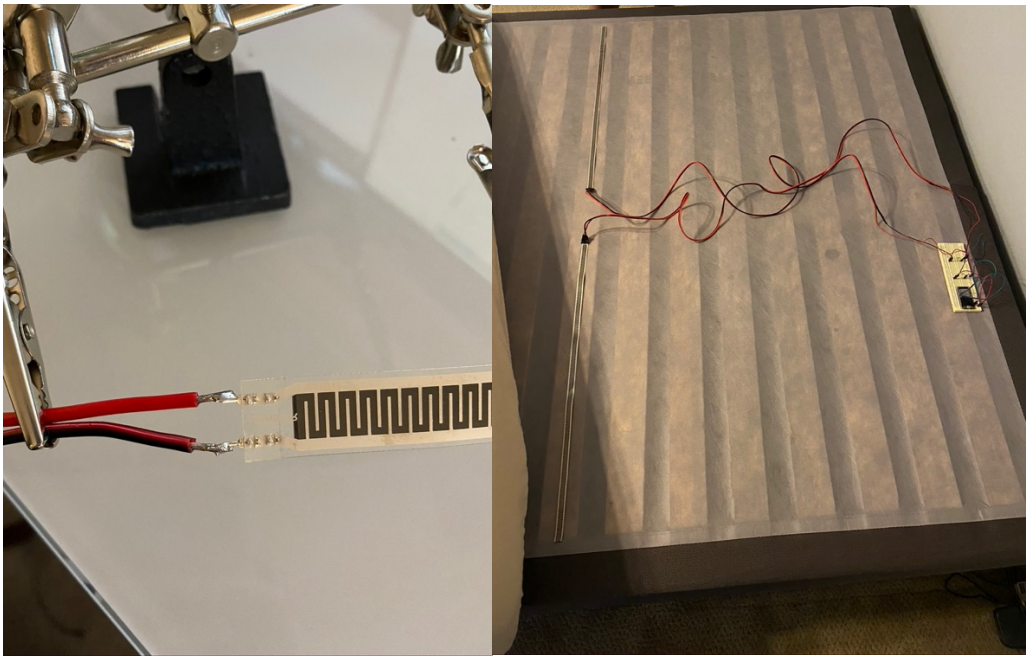
Pressure can be applied anywhere on the sensor, including multiple places, and it will register as a change in voltage to an ADC. The decrease in resistance leads to an increase in voltage

(V) in the circuit. Thus, we can measure and model the change in voltage as the change in force, where a higher voltage is equal to higher force on the sensor. It's important to note that the specific ESP32 board I'm using (DOIT ESP32 DEVKIT V1 Board) has a 12-bit SAR ADC w/values between 0 - 4095 and a sampling frequency of 6 kHz, but still struggles with low (0-0.1 V) and high (3.2-3.3 V) voltages due to the ADC itself. In this case we still get a lot of useful data, but as a result, some values could be missing (most of the time our values are over 0.1 V but there are some instances where they just register as 0 or 0.1 V). It's also important to note that while on its own, we can measure the voltage of this circuit in Volts (V), adding a secondary, constant resistor, allows us to calculate the FSR's resistance in Ohms (Ω) using the voltage divider equation. The layout of my hardware components in this sensor systems are as follows:



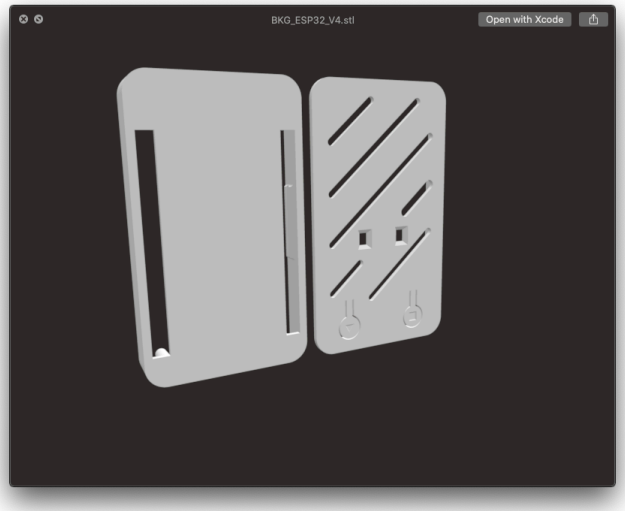
ESP32 & Sensors Hardware Configuration Diagram

This is a simple circuit in which we have two Force Sensitive Resistors powered by the 5V line and two 4.7 kΩ resistors setup and connected to ground. The purple lines each represent a respective connection to a GPIO pin.



Soldering and Installation of FSR to Bed

As for an enclosure, most would do; if developing something custom it could be rapid prototyped using a laser cutter and then printed using a 3D printer. For this task there are also models already available online which could be used to accomplish the task:



Simple ESP32 Enclosure

For this prototype stage I stuck with a breadboard, allowing me to easily prototype this solution.

Software

On the software side, one can use the function analogRead(PIN) on an Arduino (where PIN is an ADC GPIO pin), to get a voltage measurement from the FSR. After testing this works, we move to an amazing open source project called ESPHome, which allows you to flash ESP devices over the air after initial configuration (Wi-Fi updates) and use templates to determine how to use the I/O of your device. There are many useful templates here which led me to the following configuration:

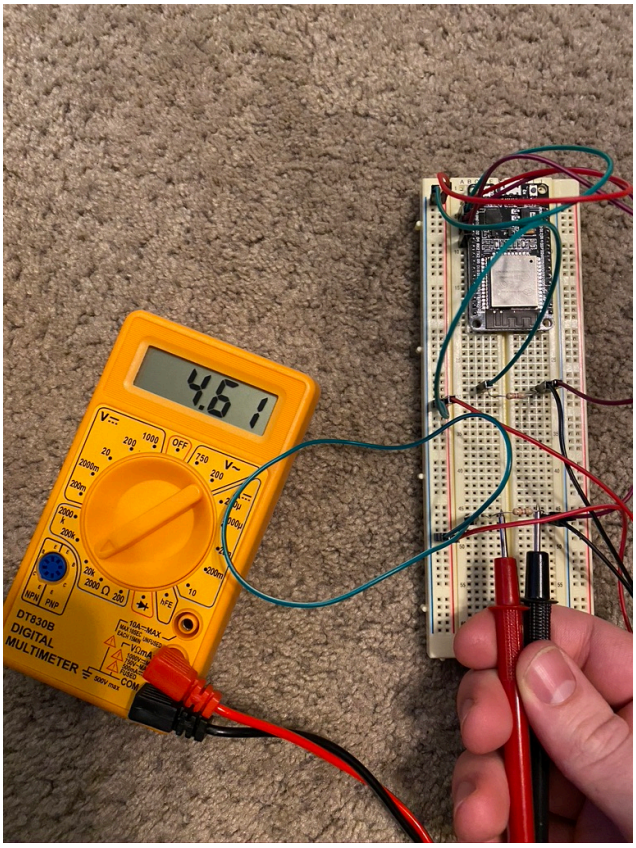
```yaml
esphome:
  name: upstairs_den_bedroom_bed
  platform: ESP32
  board: esp32doit-devkit-v1

sensor:
  - platform: adc
    id: cody_bedside_sensor
    pin: GPIO33
    update_interval: 1s
    attenuation: 11db
    name: Cody Bedside Voltage Sensor
  - platform: resistance
    sensor: cody_bedside_sensor
    configuration: UPSTREAM
    resistor: 4.6kOhm
    reference_voltage: 3.3V
    name: Cody Bedside Resistance Sensor
  - platform: adc
    id: chuan_bedside_sensor
    pin: GPIO32
    update_interval: 1s
    attenuation: 11db
    name: Chuan Bedside Voltage Sensor
  - platform: resistance
    sensor: chuan_bedside_sensor
    configuration: UPSTREAM
    resistor: 4.6kOhm
    reference_voltage: 3.3V
    name: Chuan Bedside Resistance Sensor

// Photo by Thierry Meier on Unsplash
```

ESP32 Config Using ESPHome

In this YAML file we can see there are two ADC sensors defined (one for each side of the bed). Going through the configuration, the *update_interval* means we get a reading every second. It is important to note that the frequency of updates necessary for the alarm automation is really only required for a little bit less than a minute (say 50 seconds) in order to determine whether the user is still in bed or not, but ESPHome is lightweight and I have other automations hooked up with this sensor which need to run within a couple seconds of us getting onto the bed (when I

get into bed the lights automatically turn off within five seconds). I want to point out that *attenuation* is just a value that needed to be added due to the ADC of the ESP32.

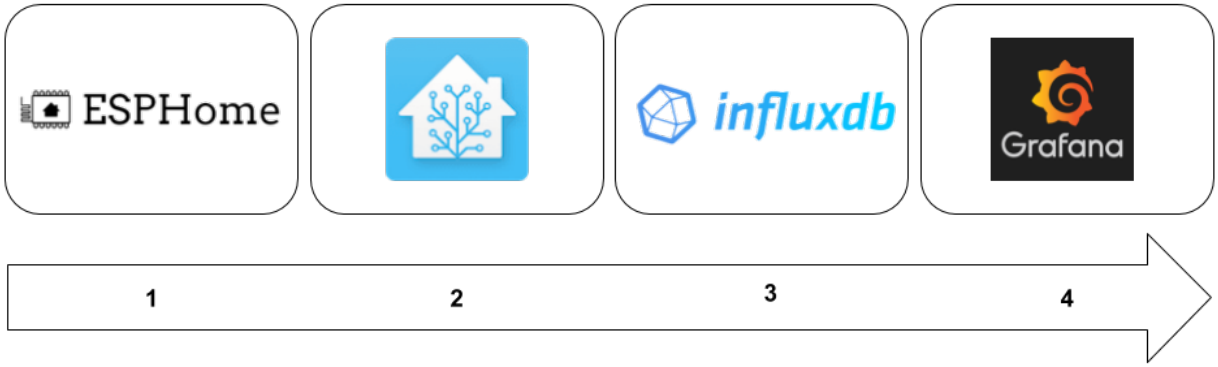
We are using these ADC sensors as input to resistance sensors as well, which are doing the voltage divider equation for us to output the resistance in ohms (Ω) of our circuit. An important note here for the software setup is that the GPIO pins are located closer to the ground than the positive end, which changes the voltage divider circuit a bit. In this case it's a simple configuration option which you'll see above is set to UPSTREAM. When I measured the resistance of each resistor it was closer to 4.6 kΩ, so I chose that as the constant resistor value or *reference_voltage* (even though I have 4.7 kΩ resistors in my circuit).



Measuring Resistance of 4.7 kΩ Resistor

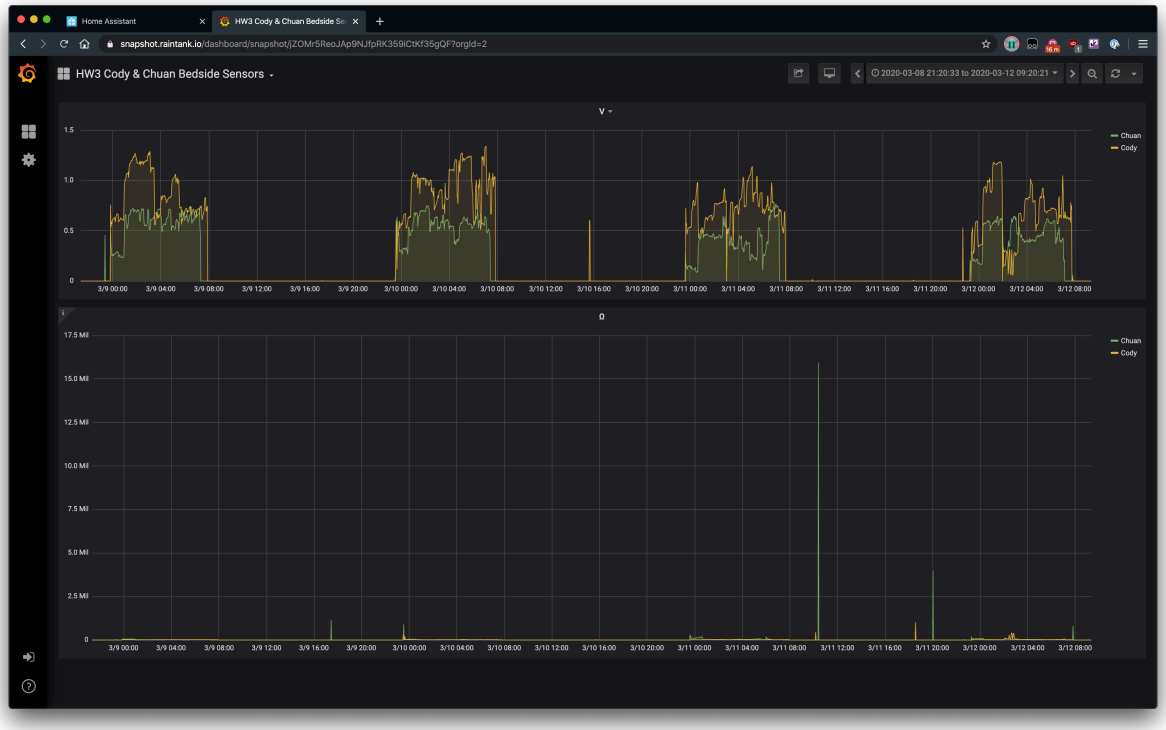
I am pushing the voltage data from this sensor into Home Assistant where it will be processed for automations. It is simple to set a threshold for the voltage detected by the ADC based on the FSR and use that to determine whether or not the bed is occupied. This can also be done on the ESP32 itself and output a simple discrete value for based on a threshold. You can also create a sliding window in ESPHome which can run on-device and do some minor processing of the data to smooth things out and get an average value. My full workflow for a delightful user

interface is as follows:



Workflow

While this is a complex set of technologies, all of it can be packaged together and setup in an easy to follow manner which may even be easier than figuring out how to flash code onto an Arduino! This is because most of the software configuration is handled already and at its most basic could just require a copy to run on another server. The technologies work as follows: **1.** ESPHome is used to get data into **2.** Home Assistant. A good time-series database is needed to store and later query the readings which are taking place every second, which is where **3.** influxdb comes in. To help graph these data for visual analysis and create a delightful user interface I use **4.** Grafana. You can view a sample of the output here and also view a snapshot web copy of some of its data here:



Grafana graphing Sensor Output

Power Output

An ESP32 consumes between 120 mA to 250 mA of current but is typically observed to run around 150 mA according to Sparkfun. For our sensor we have two GPIO connections and a direct 5V and GND connection to the sensors themselves. From the data sheet, we can see that the GPIO will consume about 20 mA of current. Multiply this by two for two GPIOs and we have 40 mA. As for the sensor, an ADC is sampling voltage and we also have a voltage divider which gives us resistance of the circuit. Since the software is doing this math for us, then using Ohm's Law, we can get the current used by each FSR. In order to get the most up-to-date information, I chose to take an average of all the data I grabbed so far inside of influxdb using the following code:

```
// Get the average voltage V:

SELECT MEAN("value") AS "mean_value" FROM "homeassistant"."autogen"."V"
WHERE time > :dashboardTime: AND time < :upperDashboardTime: AND
("entity_id"='chuan_bedside_voltage_sensor' OR
"entity_id"='cody_bedside_voltage_sensor') GROUP BY "entity_id"

// V.mean_value[entity_id=cody_bedside_voltage_sensor] = 0.69
// V.mean_value[entity_id=chuan_bedside_voltage_sensor] = 0.54


// Get the average resistance Ω:

SELECT MEAN("value") AS "mean_value" FROM "homeassistant"."autogen"."Ω"
WHERE time > :dashboardTime: AND time < :upperDashboardTime: AND
("entity_id"='chuan_bedside_resistance_sensor' OR
"entity_id"='cody_bedside_resistance_sensor') GROUP BY "entity_id"

// Ω.mean_value[entity_id=chuan_bedside_resistance_sensor] = 29.6K
// Ω.mean_value[entity_id=cody_bedside_resistance_sensor] = 26.37K


// Photo by Brady Bellini on Unsplash
```

SQL Queries to get mean voltage (V) and resistance (kΩ) over a week of data

$$I = \frac{V}{R}$$

Equation of Ohm's Law solving for current

Solving for I on each sensor for the above screenshot, we get the following:

$$I_{cody} = \frac{.69\ V}{29600\ \Omega} = 0.000023311\ or\ 2.3 \times 10^{-5} A = .23\ \mu A$$

$$I_{chuan} = \frac{.54\ V}{26370\ \Omega} = 0.000020478\ or\ 2.1 \times 10^{-5} A = .21\ \mu A$$

Solving for both FSR's Current

We can then calculate the power by calculating current times voltage and using the power equation for the sensors and the ESP32. Also note I grouped the GPIO and ESP32 currents together since I believe they're on the same 5V line.

$$P = IV$$

Power Equation

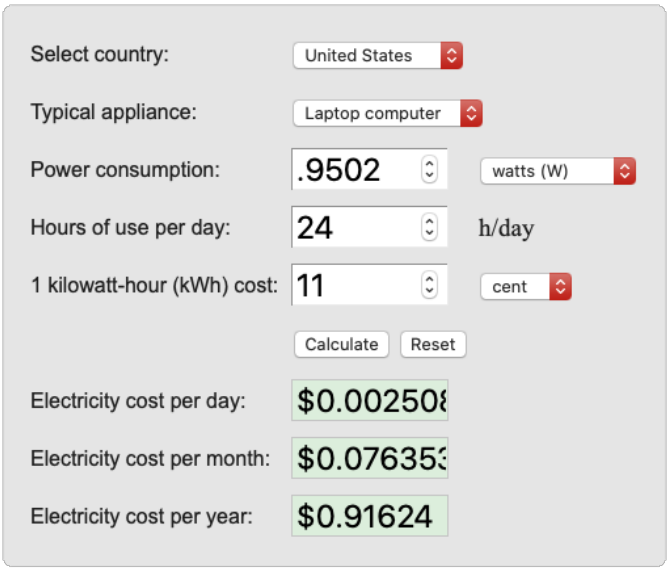
$$P_{ESP32} = (.15 + .04)\ A * 5\ V = 0.95\ W = 950\ mW$$

$$P_{FSR} = (2.3 \times 10^{-5} + 2.1 \times 10^{-5})\ A * 5\ V = 2.2 \times 10^{-4} W = .2\ mW$$

$$\sum P = 950.2\ mW$$

Figure 10: Solving for Power of FSR and ESP32

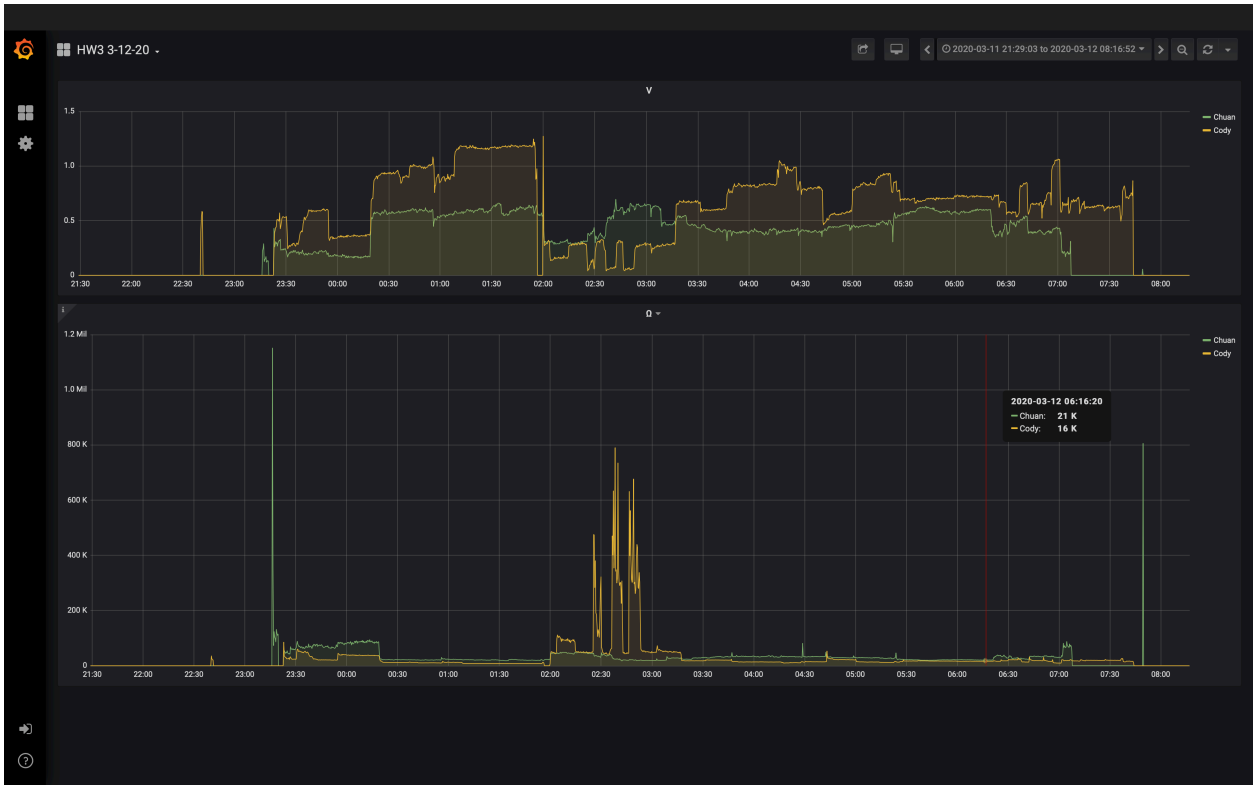
This is less than 1/60th of the power usage of a typical (60 W) lightbulb! In Seatle the current (2020) yearly electrical cost for this sensor and computer is less than $1:



Electricity Cost Per Year for Sensor System

Performance Evaluation

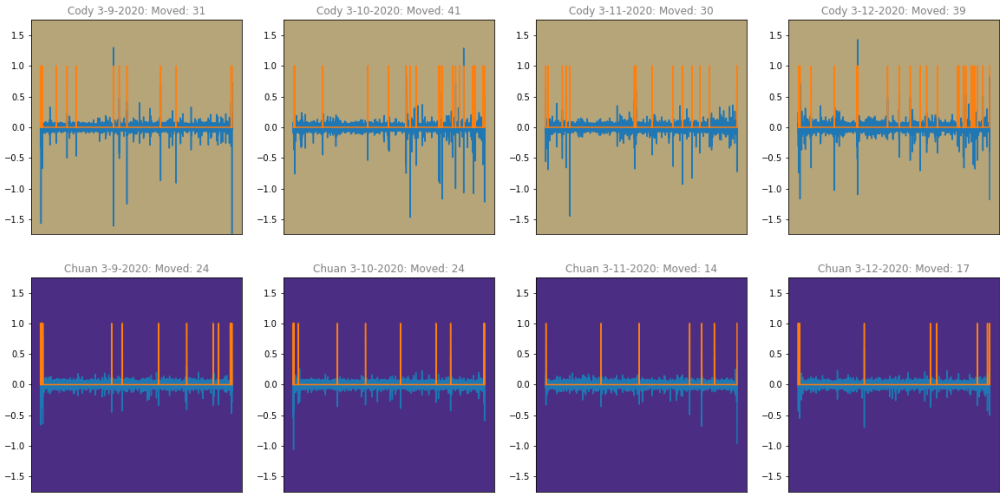
I have been using this sensor system for over a week and it's worked perfectly! Below I am including a night of sleeping data for my partner and I which can also be explored through a web browser:



FSR Graph over Night

In this graph we can see a couple instances that I think really show that it's working. When I get out of bed around 2am you see the voltage go down immediately and then resume when I get back into bed. It's really cool to think that the light turned on for me when I got out of bed and then turned back off automatically. You can also see that my partner woke up before me by about an hour. I also want to point out that there are some resistance spikes as can be seen in the graph of the Ω. This is caused by the sensor detecting the decrease in resistance to a measurable level between when the user gets on the bed and when the sensor reading occurs, as when a user isn't on the bed the resistance is effectively Infinity (∞) and is seen as 0. It's fascinating to me because this shows that the sensor is able to capture many samples of a data points within a short period of time (in this case once a second). There are also some points where the resistance increases around 2:30am and this is caused by movements and shifting weight that actually move some of the weight off of the sensor. While the software automations for waking up are still in progress, this sensor inputting to the home automation system will work to determine every minute whether a user is in bed or not.

In my other classes paper, I discovered that it's possible to use these sensors to distinguish who is sleeping in the bed based on the number of times they moved, as shown in this graph of the derivatives of these nights of sleep here:



Sensor Data Derivatives (Blue) and Threshold Values for Movement (Red)

We can see in the titles of the graphs and the graphs themselves that my partner moves less often than I do while sleeping and this can be distinguished with the readings of this sensor system using an FSR. While I previously linked to this paper in the use cases section above I will also link to it here: (must be logged into UW; if you want to read it but aren't a UW student please email me at hello@cody.codes): http://bit.ly/2INUdud.da

This sensor system satisfies the user requirements I initially set out to fulfill. I learned a lot through this exploration and creating this system and am glad I did so. I am very proud that this system works so well and am eagerly thinking about what else can be thought up to automate with it! Thank you for reading my paper!