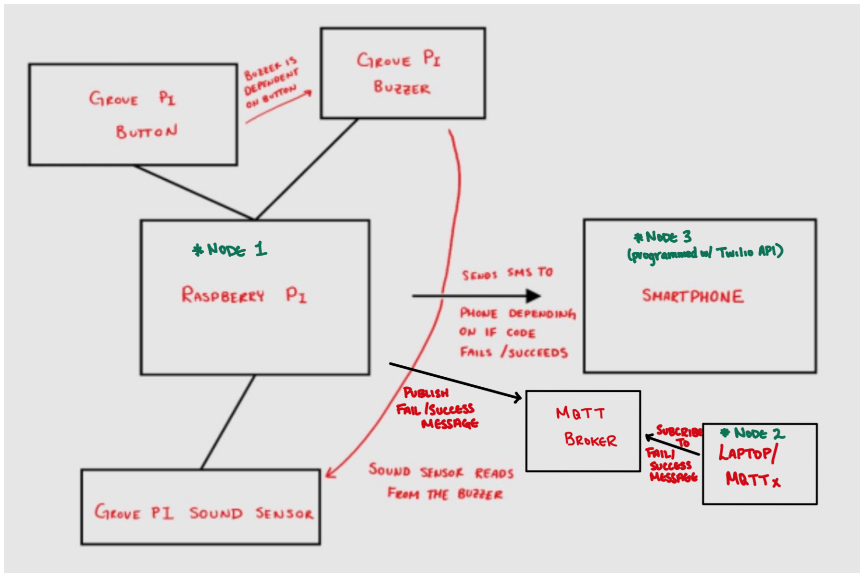
Diya Thapliyal & Pooja Kowshik

Final Project: Access Code Alert!

Description

This system is an implementation of an access-code system with an alert mechanism. This project utilizes the several concepts learned throughout the semester by utilizing a raspberry pi (along with grove pi sensors) and a smartphone to create a smart access code system that alerts the client when someone has tried to enter in the code unsuccessfully. When the user successfully enters the code, a confirmation message is sent to the user. When the code is unsuccessfully sent, another separate sms is sent to the client indicating that someone may be posing a security risk to whatever the keypad is protecting. This information is also published to an MQTT broker that a laptop VM subscribes to and the failure or success code message is displayed on a MQTTx platform. Surprise, the code is USC in morse code :0 (fight on!).

Diagram


GROVE PI BUTTON
BUZZER IS DEPENDENT ON BUTTON
GROVE PI BUZZER
*NODE 1
RASPBERRY PI
SENDS SMS TO PHONE DEPENDING ON IF CODE FAILS / SUCCEEDS
*NODE 3 (programmed w/ Twilio API)
SMARTPHONE
PUBLISH FAIL / SUCCESS MESSAGE
MQTT BROKER
SUBCRIBE TO FAIL/ SUCCESS MESSAGE
*NODE 2
LAPTOP/ MQTTx
SOUND SENSOR READS FROM THE BUZZER
GROVE PI SOUND SENSOR

Components, Platforms & Protocols

The nodes used for this project were a raspberry pi and a smartphone, with the RPi acting as the sender and the smartphone as the receiver. The Twilio API was used to provide a sending point for the SMS message being sent to the smartphone receiver. Furthermore, grove pi sensors were utilized to collect data in a chain of operations; first, the button was used to generate a sound from the buzzer, which would play as long as the button was pressed. The sound sensor picked up the noise from the buzzer (super loud!) and was processed as either a short or long input.

The protocol being used in this project is the MQTT publish and subscribe protocol. The Raspberry Pi is the publisher which sends information to the MQTT broker with a topic called "msg" when the code entered is determined to be either correct or incorrect. The VM subscribes to the "msg" topic as well and receives the failure or success of the code entered which is then displayed on a platform called MQTTx. MQTTx is an IoT visualizer that can be customized with different information sources such as the MQTT broker. We used a text box to display whether the door was locked or unlocked.

Finally for processing and visualization, our main form of data preprocessing was implementing a moving average filter with a window size of 15 data points. This was done by storing arrays of the 15 most recent data points and averaging them before rounding. This helped create a smoothing effect in the data for easier postprocessing. The number of averaged data points that were above a threshold were counted to determine whether the press was a short or long knock. The expected correct code is stored in an array and all new button presses are compared to this correct array to see if the code is being entered in the correct sequence. For visualization, depending on if the user put in the correct code, a success or failure text is sent to the client's cell phone and the laptop displays the message on the MQTTx platform.

Reflection

Some of the limitations that came up as we were developing our project were the sensitivity of the sound sensors. They do not work well with consistently loud noises due to unexpected dips back to the baseline values which is what led to us implementing the moving average filter over each window of 15 data points. This was a good example of real-life signal processing since there are often instances where the initial data being collected is not in an ideal form for the necessary processing.

Another potential limitation is the user interface and user reliability. Since we are using a Morse Code based system, we must rely on the "doorbell" users to only press the button in the accepted dot and dash patterns. A way to remediate this problem would be to create a more streamlined user interface where there could be user error warnings and correction when a user has held the button for an incorrect segment of time (too short or too long). This could be in the form of an LCD that writes out these error messages or a LED that blinks red like other common door lock systems.