Alexandra Somodi & Victoria Nunez

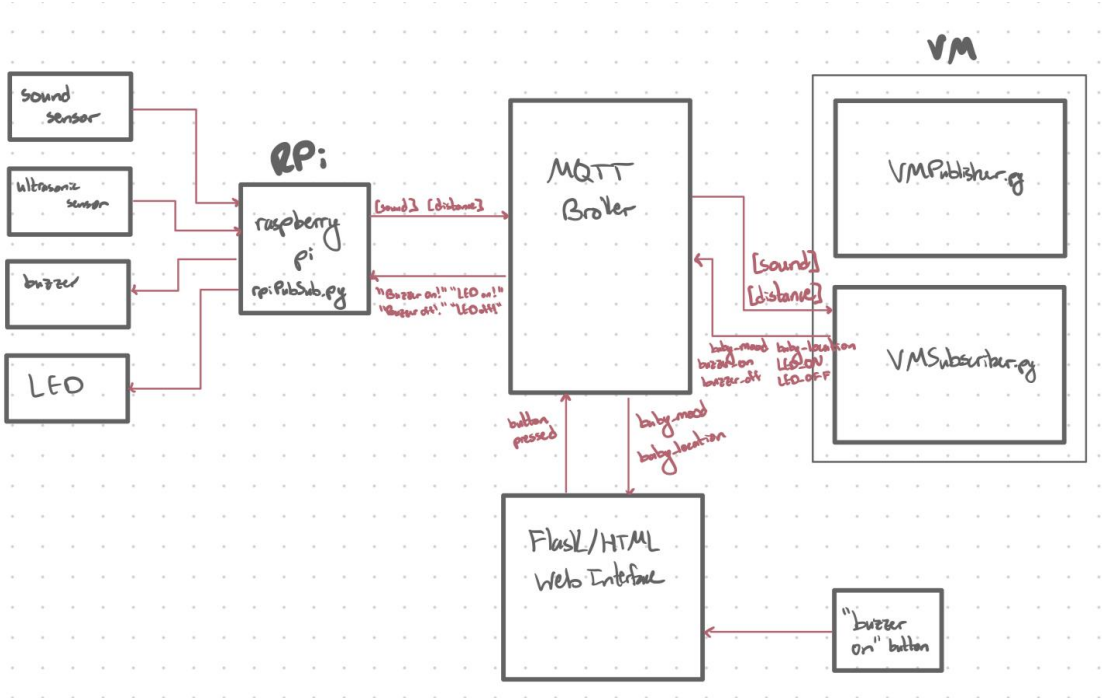
Final Project Writeup

EE 250 - Prof. Redekopp

1 December 2023

For our final project, we made the DaBaby Monitor: a fun, innovative way to keep track of your baby's mood and location! The project works as follows: it contains a Raspberry Pi, a virtual machine on a laptop, a front-end web interface, two sensors (ultrasonic sensor and sound sensor), an LED, and a buzzer. The ultrasonic sensor serves as location detection for the baby – if the baby is more than 20 centimeters away from the sensor, the LED will light up. Meanwhile, the sound sensor serves as mood detection – if the baby surpasses a sound threshold of more than 400, this is considered crying, and the soothing lullaby of the buzzer will play to calm the baby. Along with the buzzer/LED response, the baby's status is displayed on the front end web interface. One box displays "Baby's Location" with two options: "Baby is close" or "Baby ran away!" Another box displays "Baby's Mood" with two options: "Baby is calm" or "Baby is crying." These statuses update upon refreshing the page. The web interface also contained a button labeled "Play Buzzer!" As the name suggests, upon pressing the button on the website, the buzzer would play for one second.

Our system can be visualized through the following block diagram:



This project utilizes primarily MQTT for its communications and interactions, but we also used Flask to help build our web interface. The ultrasonic sensor and sound sensor collect data from the baby's surroundings every second; this data is processed using an ADC. Once the RPi receives it, this sensor data is published to two topics: asomodi/sound and asomodi/ultrasonic. That information gets sent to the MQTT Broker, who sends it to the VM Publisher / Subscriber. The VMSubscriber.py then decides if the baby is crying or calm, and if it is near or ran away. After the status of the baby is determined (using simple conditionals), a command to turn the buzzer on/off and LED on/off is sent to the MQTT broker and then to the RPi. The RPi receives the message, and performs the action requested by the VM. Meanwhile, a baby status update is sent in baby_mood and baby_location to the MQTT broker and then FlaskApp.py. The Flask app stores that information, and once the web interface is refreshed, the baby's status is updated. If the "Play Buzzer!" button is pressed on the web interface, a message is published to buzzer_command, and sent to the MQTT Broker. That information is then sent to the RPi, which turns on the buzzer. We decided to use MQTT since it would be the easiest to divvy up communications based on inputs, outputs, and specific requests three ways (VM, RPi, and web interface) going either way.

One limitation of this project is that the web interface wouldn't update unless refreshed. It would be really cool to just have it running live all the time, and probably more practical than requiring a refresh in a real-life scenario. Also, the ultrasonic range sensor is probably not the best way to detect if your baby is still present or not. What if it just is lying a weird way in the crib? Suddenly you think your baby is missing? It's not helpful to have inaccurate data posted to your device. Through this project, we learned a lot about how to approach a full-scale, open-ended project. A good chunk of planning has to go into deciding how the system should look, what components to use, how to get them to interact, etc. It makes actually *starting* the project one of the most difficult aspects of it. Additionally, we learned a lot about troubleshooting, since we had some integration issues! Nonetheless, overall, this project was a really great learning experience and a lot of fun to put together.