Josie the Office Assistant

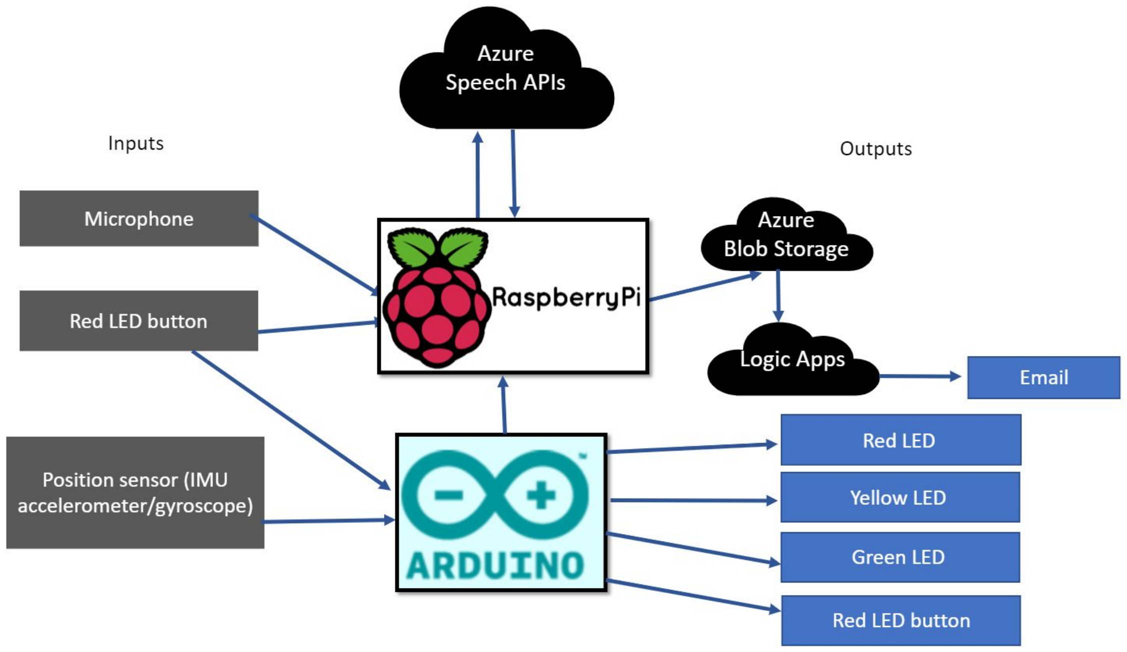
Never miss another co-worker when in a meeting

Project Description

The Office assistant is a cube-like structure that can be placed on a desk. We have chose to divide the project into two parts. The first part of the project was centered around displaying whether or not the user is free to communicate with co-workers. When the user changes the position or orientation of the cube a specific LED color is shown to indicate whether the user is: available, busy, etc. We utilized an accelerometer and ball-tilt to detect the positioning.

The second part of the project, is to allow someone visiting an office to leave a message in the event the person they were looking for is absent. We used a LED button to start the voice recording which stopped recording after 15 seconds. The voice messages are captured using a microphone. The LED button also provided a visual indication when pressed to indicate a recording is being made. Once the recording is complete we leveraged Microsoft Azure Cognitive Services' speech APIs to transcribe the voice into text. Then an email is sent with the original audio file and the transcribed message to the Office assistant owner using Logic Apps.

How it works



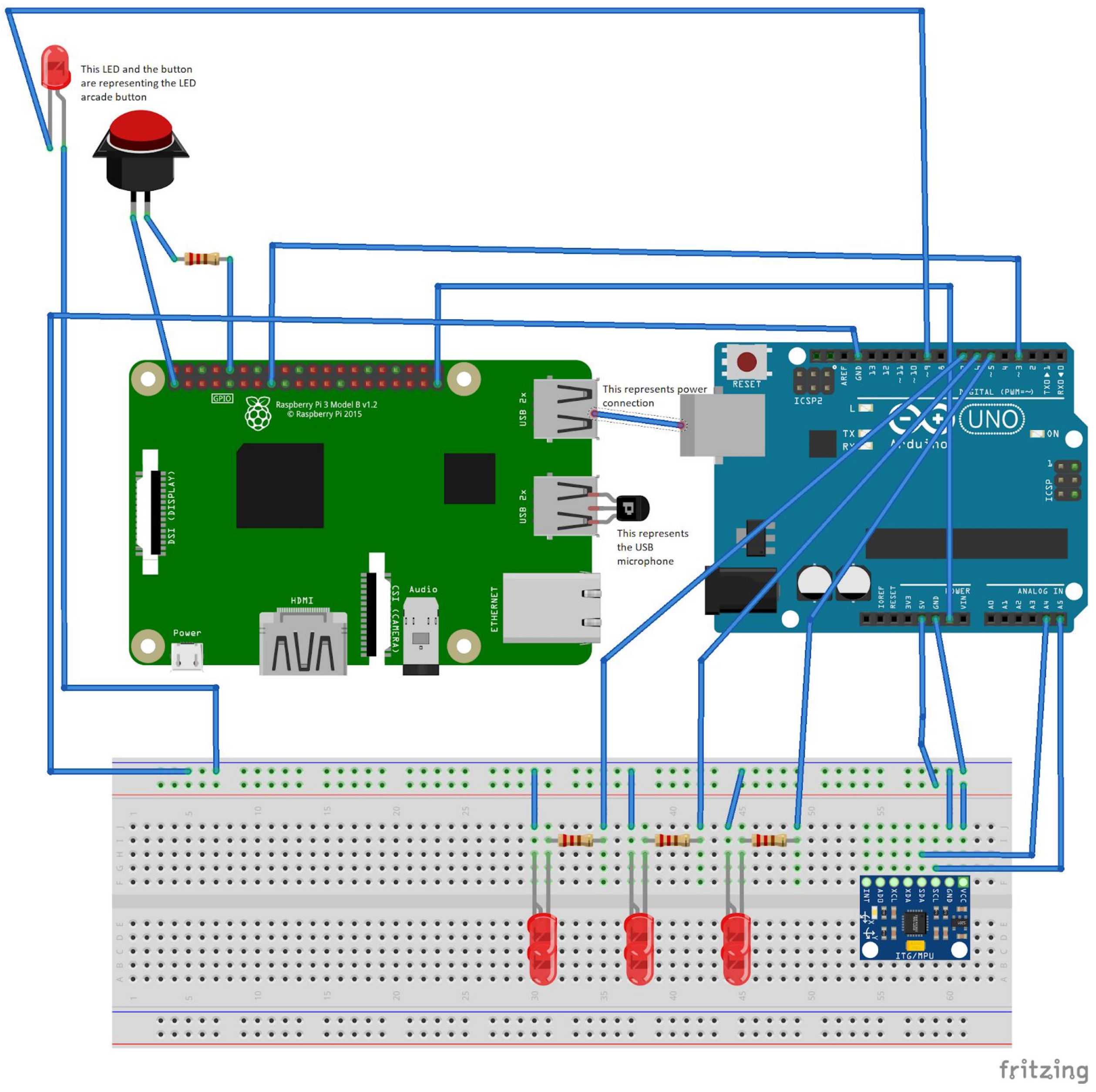
Implementation

The project is divided in two major components: Position-sensing and Voice Recording. The position-sensing portion of the project is managed by the arduino board. For this portion of the project we needed a MPU-6050 accelerometer/gyroscope and 6 LEDs, 2 red, 2 yellow, and 2 green. We used the accelerometer/gyroscope to determine position readings and return this to the Arduino to compute the orientation of the Office Assistant. This computation was done leveraging a formula from an Arduino discussion about the same model of the accelerometer. We made some adjustments to the formula in order to ensure we could map the positions to specific sides of the cube. Once this was done we mapped each LED color to a single side so a single color would be shown on 1 of 3 sides.

The voice recording portion of the project is managed by the Raspberry Pi. For this portion of the project, we needed an LED arcade button and a USB microphone. The microphone is used to record the messages we send to the Azure and transcribe using the speech APIs. The arcade button contains a physical button circuit and a LED circuit. The physical button circuit is connected to the Raspberry Pi through a GPIO pin and ground. When pressed it initiates the message recording and a message is sent to the Arduino (which is powering the LED circuit of the button) through a Raspberry Pi GPIO pin. This was done because the LED required 5V of power. When Josie is not recording, the button fades on and off slowly but once the button is pressed the LED begins flashing quickly to indicate recording is in progress.

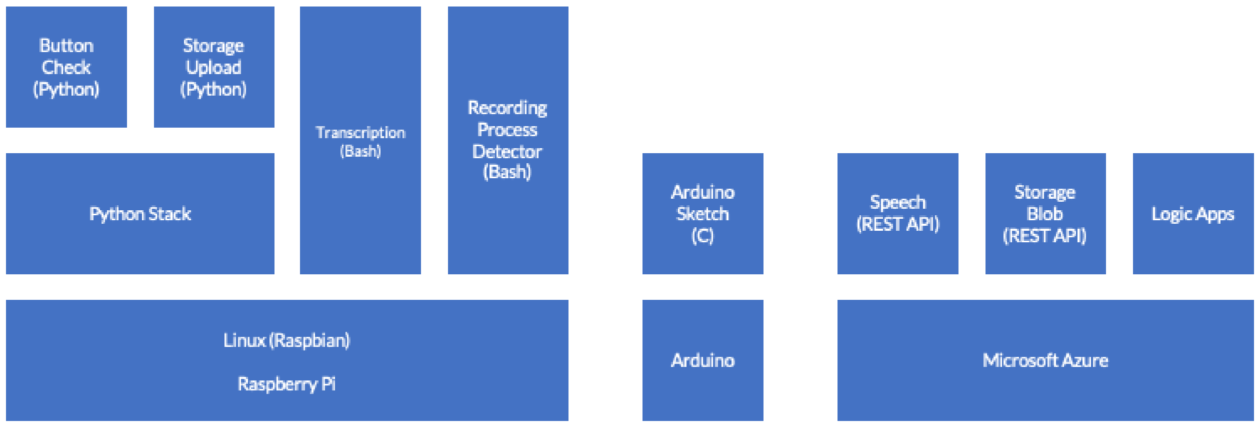
Each recording session lasts approximately 15 seconds which managed using and once the recording is stopped the Raspberry Pi sends another message to the Arduino so the LED button light can change state once again. The last piece of this portion is the message transcription and emailing. There is a process continuously running on the Raspberry Pi that detects when a new recording is created and then creates a CURL request to the Azure Speech API to get the transcription and store it as a text file on the Raspberry Pi. Once the transcription is created, both the recording and the text file are sent to Azure Storage, for archiving, and Azure Logic Apps, to automate the sending of an e-mail to the owner of the office assistant.

Circuits


This LED and the button are representing the LED arcade button
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015
This represents power connection
This represents the USB microphone
UNO
ITG/MPU
fritzing

Software

Below we show the main software components used and discuss them in more detail.



Software on the Raspberry Pi:

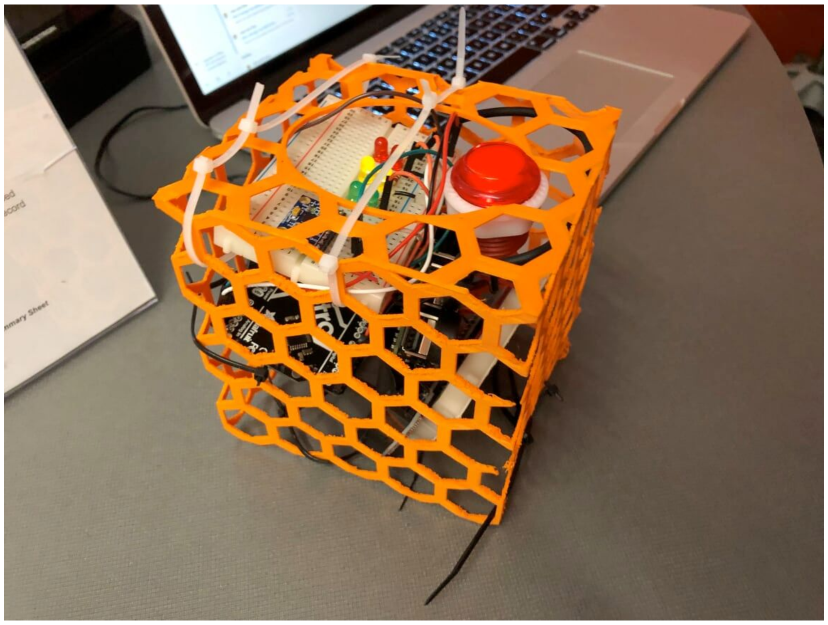
1. **Button Check (buttonprod.py):** Used to initialize the GPIO mode in order to properly detect the arcade button presses, and initiate the recording when pressed.
2. **Storage Upload (folderwatcherprod.py):** Used to detect when a new recording is made, call the transcription process, send the output of the transcription to a text file, and upload both the recording and transcription to the Azure Storage blob.
3. **Transcription (transcriptionprod.sh):** Used to create a CURL request that sends the recording to the Azure Speech service.
4. **Recording Process Detector (processcheckprod.sh):** Detects when a recording is happening, and kills the process after 15 seconds and restarts the recording button by killing and restarting the button check program.

Software on the Arduino (running in parallel to the Raspberry Pi):

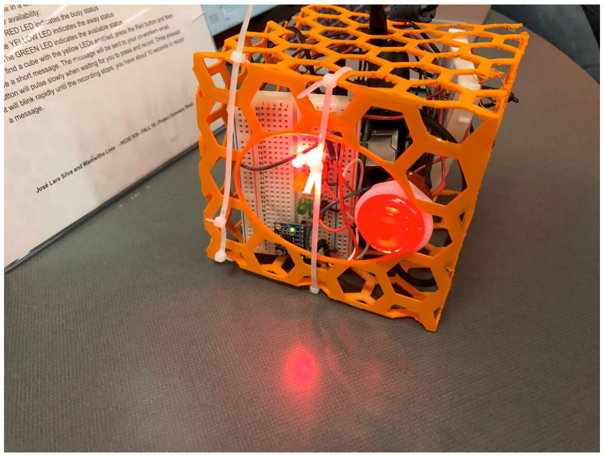
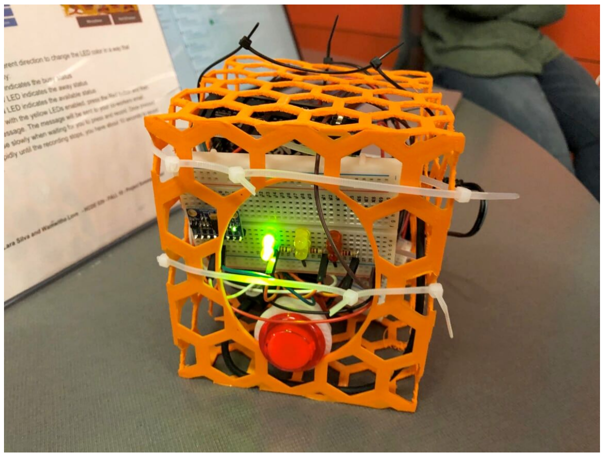
1. **When powered on:** Initiate the accelerometer, LEDs, the arcade button LED, and the recording state input pin (connected to the Raspberry Pi).,
2. **Always running:**
    a. **Position detection:** Read the accelerometer/gyroscope values, calculate an estimated range of values to determine the current orientation and turn on/off a specific set of LEDs
    b. **Recording state:** Use a digital read to get the current recording state from the pin connected to the Raspberry Pi. Then turn the LED on the arcade button on or off.
        i. If the recording mode is not active, increase the brightness progressively until max value is reached, then decrease until min value is reached.

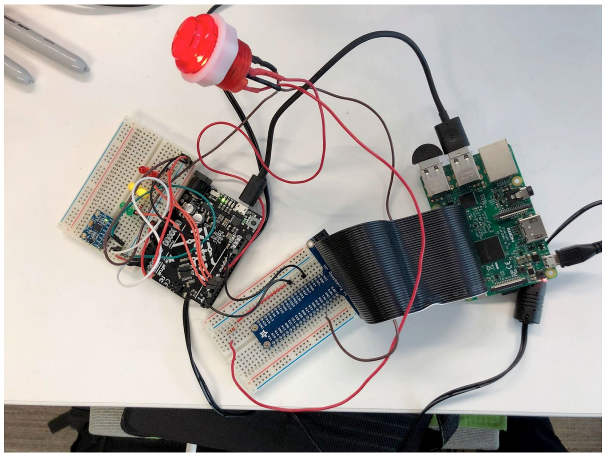
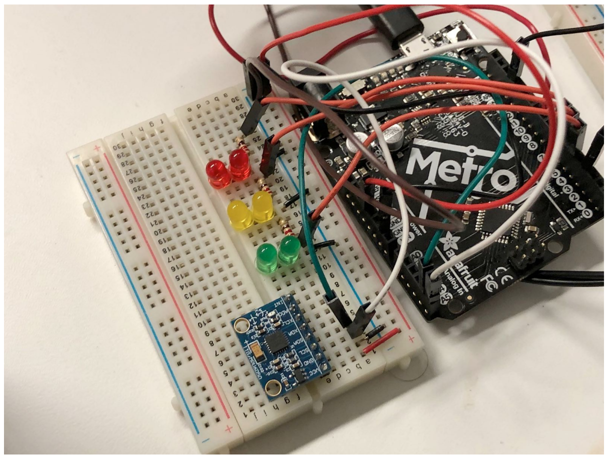
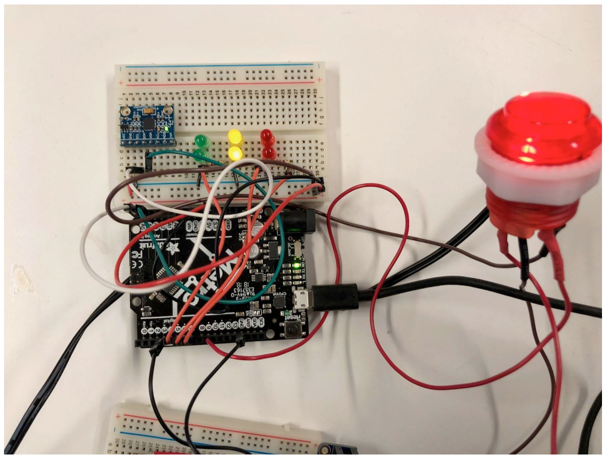
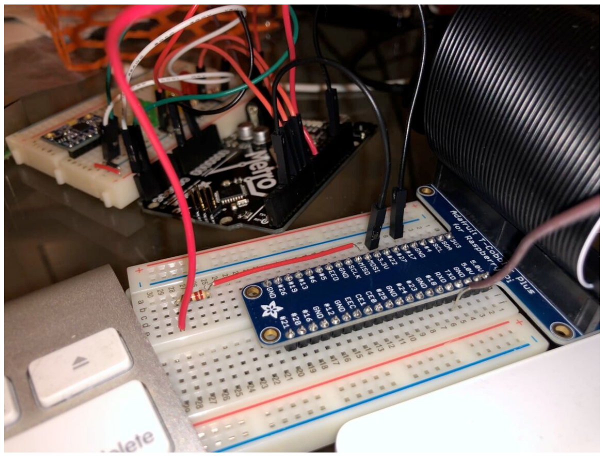
ii. If the recording mode is active, turn the button's LED to full brightness, then to zero brightness.

Gallery



Josie - The Office Assistant in their full glory [See it in action!](#)
Note: The yellow LEDs are on indicating the user is not available so the visitor can leave a message by pressing the red LED button.

 Tilted to the side: Red LEDs turn on indicating that the user is not available.	 Tilted to a different side: Green LEDs turn on indicating that the user is available.

 Overview of the entire solution outside its enclosure.	 Overview of the Arduino and accelerometer/gyroscope parts.
 Overview of the Arduino part including the LED in the arcade button.	 The Raspberry Pi side, using a Pi cobbler, the 3.3v and resistor connected to the button closing the circuit on the RXD pin.

Process

Overall, the project was more challenging than we expected. We had many challenges along the way and had to think through different work arounds. Some examples of these work arounds are: the arcade button LED state and the use of the Raspberry Pi to record the voice messages. Ironically, the addition of the Raspberry Pi was to simplify the use of the microphone.

The arcade button was an unexpected challenge because we wanted to be able to change the LED lighting based on recording state. In the end, the we had to plug the LED portion of the button into the Arduino and leave the button plugged into the Raspberry Pi. While it worked out in the end, this ended up requiring us to find a way to communicate state between the Arduino and the Raspberry Pi.

Introducing the Raspberry Pi introduced considerable dependencies, such as hardware to debug when things did not work properly, and a need to set-up the entire solution to connect to Wi-Fi each time. Towards the end, this project became a Linux learning experience; from the way processes are handled, to understanding how to automatically identify processes running in the background and kill or respawn them without any user intervention. These specific challenges were quite painful to overcome.

Attempting to create a real-world physical prototype although challenging held many delights throughout the process such as: learning to use a 3D printer and all the small wins throughout the process. It seemed as each component came online we had a new joy to celebrate. In the end, the hours we put in yielded a final version that while not exactly what we had planned still was able to function as an Office Assistant.

Additional References

- Azure sample for the calling the Speech APIs
- Azure docs for uploading files to Storage Blobs using Python
- Connecting a button to Raspberry Pi using GPIO libraries
- Reading button clicks and debouncing in Raspberry Pi
- Using Watchdog in Raspberry Pi to detect folder changes