CSC 591 - IOT HW 4 - [G16]

INTRODUCTION

We have connected the MPU6050 sensor to a Raspberry Pi and interfaced using Python to read the values from the Gyroscope and Accelerometer from the MPU6050 module. It uses an I2C bus and 4 wires. The circuit diagram for this connection is shown on the image below:

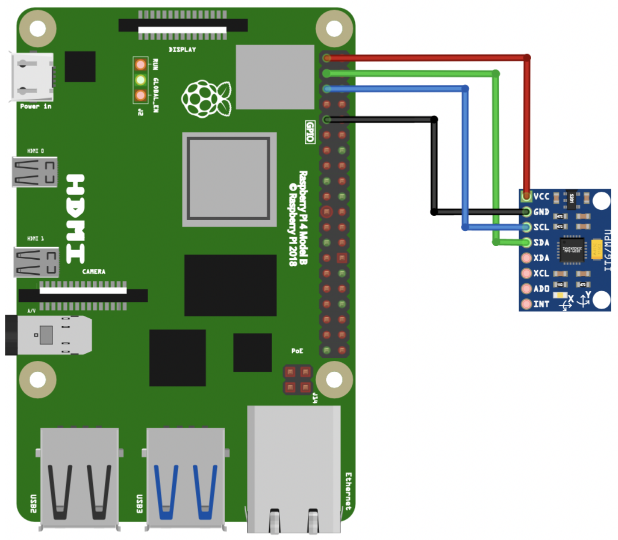


Figure 1: Door RPi and IMU Schematic

The mounting of the IMU and orientation is shown on the images below:

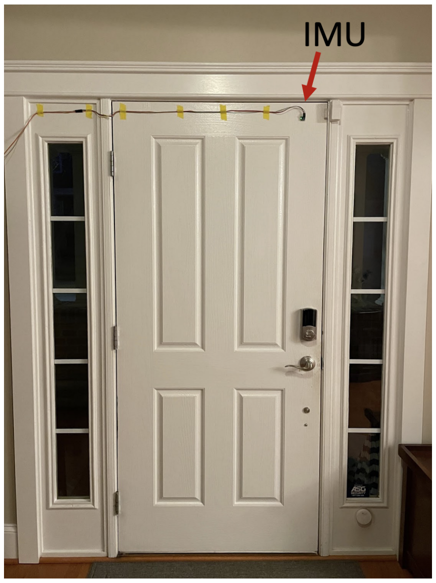


Figure 2: IMU Door Integration

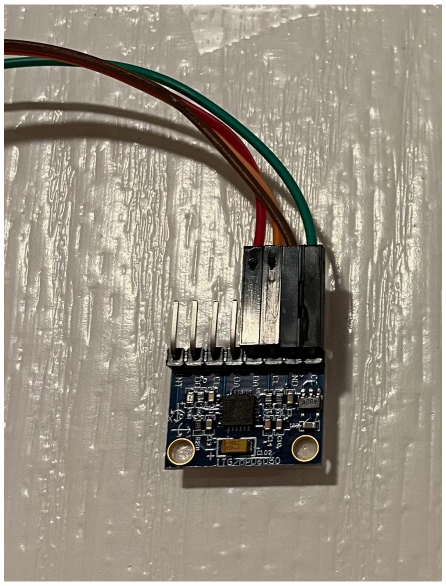


Figure 3: Door IMU Mounting Orientation

HARDWARE DESIGN

The Hardware Design consists majorly of an MPU6050 sensor, Raspberry Pi 4 (IOT device), a Laptop running as a cloud client to handle classification with machine learning and a third Laptop running the cloud monitor client to record and display the results of the classifier. Additionally, the IBM Watson IOT platform acts as our MQTT broker.

MPU6050 Sensor - which contains a 3 axis accelerometer and a 3 axis gyroscope, 3-axis accelerometer and a digital motion processor, all in a single IC to sense the inertial motion of the door. The RPi communicates with this module using I2C communication protocol. Various parameters can be found by reading values from addresses of certain registers using I2C communication. It monitors the status of the door whether it is open or closed. Data was synchronously sampled at 100 Hz. This rate provides a detailed signal while remaining within the limits of data transfer capability between the MPU6050 and the raspberry pi 4.

Raspberry Pi - This is used to process the data it receives from the sensor. MPU6050 is communicated with Raspberry Pi through the I2C data bus at the clock frequency of 400kHz. Figure 1 shows the breakout board of MPU6050. For collecting the results, we have run a classification algorithm on the IOT device. The technique and implementation of the same is described in the software design section below.

Classifier Laptop - This device will subscribe to 'imu' topic. Whenever the Raspberry Pi attached in the door sends data to the broker (Watson IoT Platform), it is redirected to this device. The data received will be processed by the classification algorithm implemented in this device and the result will be sent to the monitor device as a command.

Monitor Laptop - The IOT device will send the information to an application on the laptop. This application will continuously show the current status of the door. Both the IoT device and the laptop/smartphone will be MQTT clients. IoT devices will be published. Laptop/smartphone will be the subscriber.

SOFTWARE DESIGN

The software design consists of 3 modules. First is the software that monitors the door IMU, captures motion events and publishes the data stream from IMU to the cloud (Door Event Detection / RPi). Second, a classifier which subscribes to the 'IMU' event, runs a classification algorithm on the received data and sends the result to the monitor device as a command (deployed in a server). Third, a monitor device which receives the data from the classifier and displays the final door state in the terminal (laptop).

Door Event Detection - To accurately detect motion events the code executes an auto calibration procedure when first executed. The auto calibration procedure works as follows:

1. Wait for a 'steady state' buffer to fill up and the result to indicate the door is not in motion. This is done by checking for a delta between the max and min values in this buffer to be under a threshold.
2. While steady, the values of the IMU registers are captured for a defined amount of time. Note, if during the capture phase, the door senses motion, it will revert to step 1 above.
3. After recording the required amount of data, the average values of the registers are used to zero out (i.e. tare) the values. The standard deviation of the Gx signal is used to accurately start and stop the motion capture events.

Once the auto calibration is complete, the door enters an event monitoring mode. Events are captured by monitoring the Gx value as follows:

1. When a Gx value is seen above a defined number of standard deviations, start a timer to consider this motion.
2. If the Gx value remains above the threshold until the timer expiries, consider this a motion event and start capturing event data. If the Gx value does not remain above the threshold during the timer, then this is not considered a motion event and is not captured
3. Once a motion event has started, continue to capture the event data until the Gx value drops below a defined number of standard deviations and debounce this threshold similar to step 2 above.
4. When a motion event has been captured to completion (see Figure 4 below for an example of a captured event), a flag is set that is used by the higher level software to publish this information to the cloud broker. Note: The software also exposes the triggered start and end of the motion used when training and classifying the event.

One other use of the door event detection code is to capture motion events for training purposes. If configured in the training mode, events and associated data (i.e. motion start and stop indices) are instead written to individual .csv values and 'classified' using a simple heuristic.

Classifier - We used a convolutional neural network (CNN) to classify door opening/closing using the MPU6050 Gx signal. We went with this approach because in the past, Radek had success using this type of model for classifying human activity (walking on grass vs concrete, walking upstairs vs downstairs) with accelerometer and gyroscope signals. This model was built using TensorFlow in python.

Upon analysis of the 6 output signals, it was found that the Gx signal contained enough information to classify the door events.

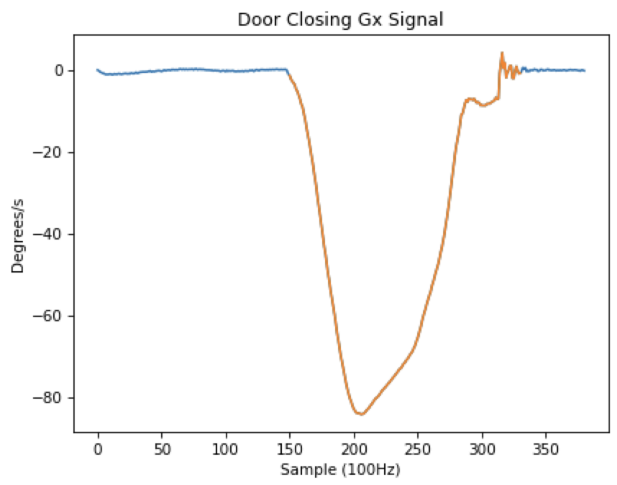


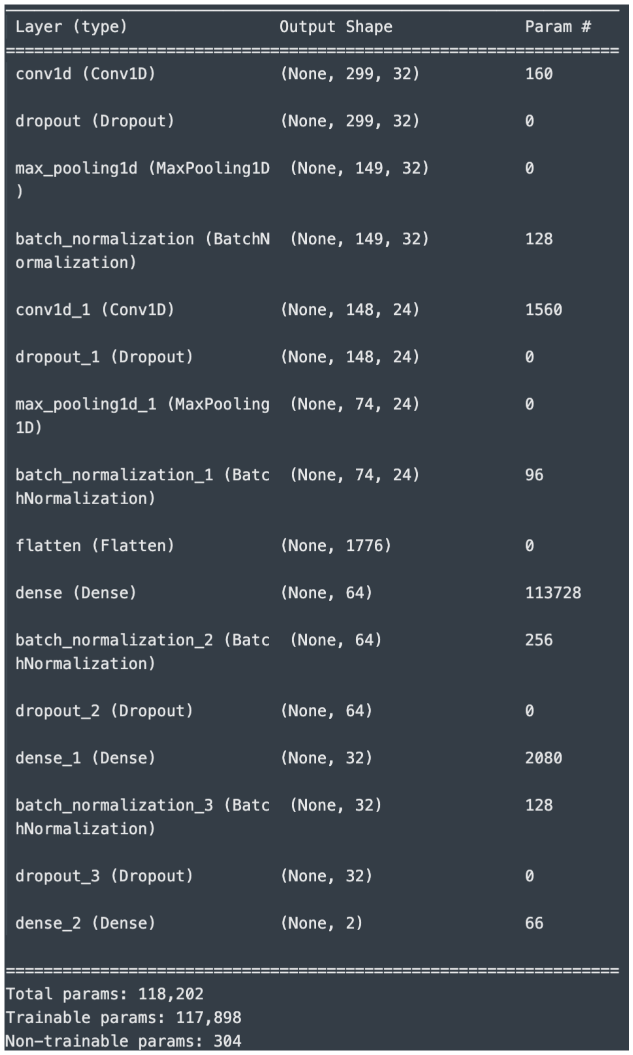
Figure 4: Door Closing Gx Signal

Shown here is a Gx signal for closing a door. The full collected signal is in blue in orange, while the orange portion corresponds to the true door open start and stop. The blue is the extra time before and after the event. Opening a door generates a similar waveform, but positive instead of negative as the door swing direction is reversed. Analysis of the other channels yielded either weak or noisy signals.

The model was designed to take the input signal from the true start/stop of the event. Before entering the model, data is preprocessed. Preprocessing steps include:

1. Downsampling the amount of data points, if there are more than 600. Samples are downsampled by a factor of 2, if this is not enough, then the factor is increased by 1 until the amount of data points is less than or equal to 600.
2. Normalize the data by subtracting the mean of the signal from each data point and dividing by the standard deviation of the signal.
3. Finally, 0s are appended to the data until the number of data points is equal to 600.

It is important that the number of data points is 600, as this is the required shape of the input into the model. This corresponds to about ~6 seconds of data at 100Hz. This is a long time that should cover most door events at the full 100 Hz sampling rate. Additionally, normalizing the data assists with smoother model training.


```
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 299, 32)           160

 dropout (Dropout)           (None, 299, 32)           0

 max_pooling1d (MaxPooling1D  (None, 149, 32)          0
 )

 batch_normalization (BatchN  (None, 149, 32)          128
 ormalization)

 conv1d_1 (Conv1D)           (None, 148, 24)           1560

 dropout_1 (Dropout)         (None, 148, 24)           0

 max_pooling1d_1 (MaxPooling  (None, 74, 24)           0
 1D)

 batch_normalization_1 (Batc  (None, 74, 24)           96
 hNormalization)

 flatten (Flatten)           (None, 1776)              0

 dense (Dense)               (None, 64)                113728

 batch_normalization_2 (Batc  (None, 64)               256
 hNormalization)

 dropout_2 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 32)                2080

 batch_normalization_3 (Batc  (None, 32)               128
 hNormalization)

 dropout_3 (Dropout)         (None, 32)                0

 dense_2 (Dense)             (None, 2)                 66

=================================================================
Total params: 118,202
Trainable params: 117,898
Non-trainable params: 304
```
Figure 5: ML Model Summary

The structure of our model is shown in the figure above. There are 2 convolutional blocks, followed by 2 fully connected blocks and a final out layer containing 2 fully connected nodes. A single convolutional block contains the following layers: convolutional (conv1d), dropout, max pooling (max_pooling1d) and batch normalization (batch_normalization). A single fully connected block contains: a fully connected layer (dense), batch normalization and dropout layer.

This model architecture was taken from the previous experience and reduced down to prevent overfitting. Dropout and batch normalization help to prevent overfitting. Data was collected from Radek's and Justin's devices to create a dataset. Then random sampling was used to create a training, validation and test set. The split was 64%, 16% and 20%, respectively.

Data augmentation was used in order to generate additional training data. This was mainly done to train the model in case the event detection algorithm was early or late in determining when the door event occurs. This was performed by capturing additional data before and after the door events. Then we would include up to 1s additional data before the event and cut up to .3s from the main signal.The loss function for training was cross entropy loss. Shown below is the learning curve and model accuracy for the training and validation sets.

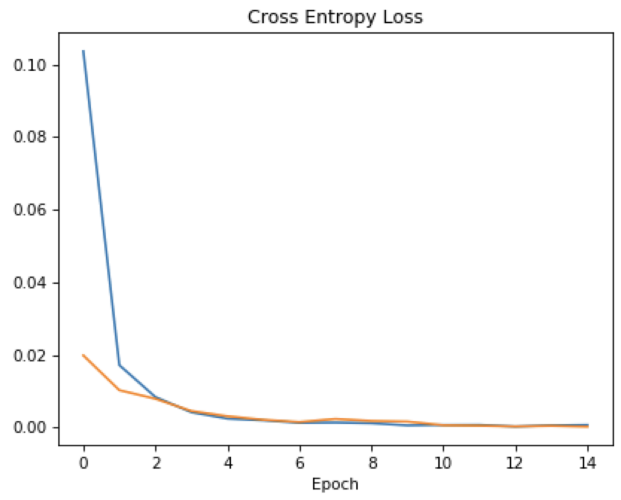


Figure 6: ML Training Loss Curve

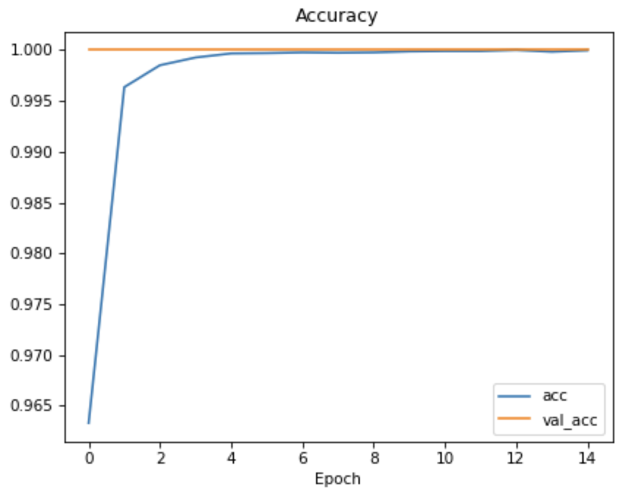


Figure 7: ML Training Accuracy Curve

We selected the epoch with the best performance (100% accuracy on the training and validation sets). We also tested with an additional test set and achieved 100% accuracy. Experimentally, this model was shown to be robust, capable of handling even minor door openings and closings (5 degrees) as well as wide door opening (>90 degrees), all at various speeds.