IoT Distance Measurement and Alert System with Adafruit IO Integration

Gustavo Kang Shim

T00659719

Computer Science

Thompson Rivers University

September 23rd, 2023

Anthony

Computer Science

Thompson Rivers University

805 University Drive

Kamloops, BC, V2C6N2

Table of Contents

Chapter 1 - Project Overview

Objective

The objective of this project is to make utilization of the HC-SR04 Ultrasonic Sensor, Adafruit IO (The interface that connects a Raspberry Pi component to a Dashboard), and a buzzer. With the components, develop a script to measure the distances that the Ultrasonic sensor captures and utilize this to measure the distance, alert based on thresholds and display a real-time distance in the UI itself.

Context and Significance

The Internet of Things (IoT) has become an integral part of modern technology, offering a vast array of applications that impact our daily lives. This project aligns with the significance of IoT, as it demonstrates a practical use case where IoT devices can enhance automation and control in various environments.

In this project, we leverage the capabilities of an HC-SR04 Ultrasonic Sensor to collect real-time distance data. On top of that, we further optimized it and included this in the Adafruit IO, sending data straight to its dashboard. We use this data to control the Buzzer's beep alerts based on the thresholds. The significance of this project lies in its real-world applications, including:

- Smart Parking Management: Use the HC-SR04 sensors to monitor parking space availability in a parking lot. The real-time data can be sent to an application or digital signage, helping drivers find vacant parking spots quickly.

- Industrial Equipment Safety: Implement the system in an industrial setting to monitor the distance between moving machinery and human operators. If someone gets too close, the system can trigger alerts and halt machinery to prevent accidents.

- IoT-Based Home Security: Utilize the sensor to detect the presence of individuals near entry points of a home. It can be part of a home security system that alerts homeowners of any unauthorized access attempts.

- Waste Bin Management: Install the sensors in waste bins to monitor their fill levels. When the bins reach a specific capacity, the system can trigger alerts to waste management companies, optimizing collection routes and reducing unnecessary pickups.

Key Components Needed:

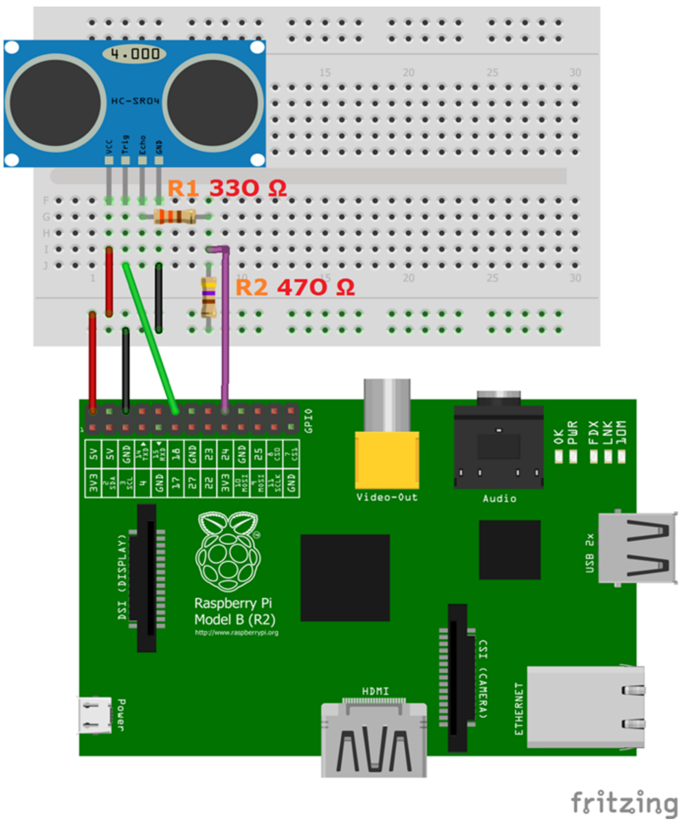
- Raspberry Pi
- HC-SR04 Ultrasonic Distance Sensor
- Adafruit IO account
- Jumper wires
- Resistors (330Ω and 470Ω)
- Buzzer

Project Description:

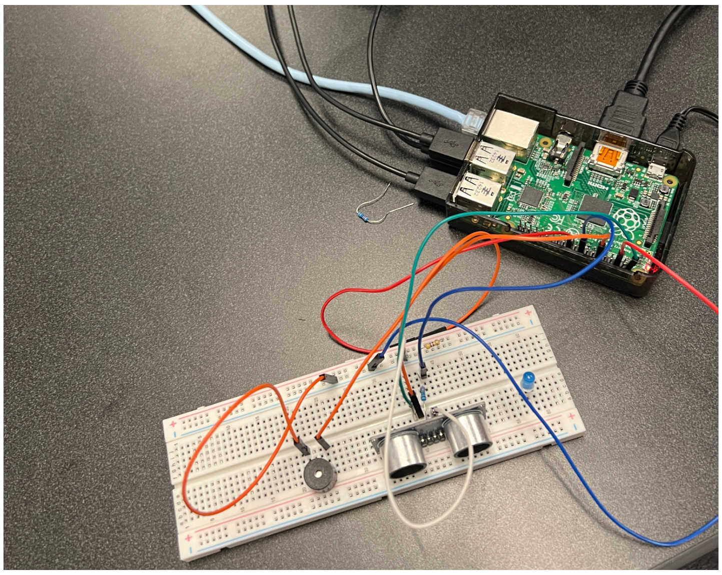
This project involves a Raspberry Pi and an HC-SR04 Ultrasonic Distance Sensor to measure distances. The measured distance values will be transmitted to Adafruit IO, where we can monitor and visualize the data in real-time on a custom dashboard. Additionally, we will set up a buzzer alarm to respond to specific distance thresholds.

1. Setup the Hardware:

- Connect the HC-SR04 module to the Raspberry Pi as follows:
  - VCC to Pin 2 (VCC) on the Raspberry Pi
  - GND to Pin 6 (GND) on the Raspberry Pi
  - TRIG to Pin 12 (GPIO18) on the Raspberry Pi
  - Connect a 330Ω resistor to the ECHO pin of the HC-SR04 module.
  - Connect the other end of the 330Ω resistor to Pin 18 (GPIO24) on the Raspberry Pi.
  - Connect a 470Ω resistor to the same end of the 330Ω resistor.
  - Connect the other end of the 470Ω resistor to Pin 6 (GND) on the
  - Raspberry Pi.



- Connect a buzzer to an available GPIO pin on the Raspberry Pi.



2. Adafruit IO Setup:

  - Create an Adafruit IO account if you don't have one.
  - Set up a new feed on Adafruit IO to receive distance data.
  - Create a custom dashboard and add widgets to display distance measurements and control the buzzer.

3. Programming:

- Modify the existing Python script to include Adafruit IO integration.
- Use the Adafruit IO Python library to send distance data to the feed.
- Implement logic to trigger the buzzer based on predefined distance thresholds.
- Set up distance thresholds for "Far" and "Close" alerts.
- Configure the buzzer to activate when the distance falls within these thresholds.
- Customize the buzzer's behaviour, such as the sound pattern or duration.

```
"""
Oct 10th, 2023

Authors: Gustavo Kang Shim, Zion Chong, Gregorson Mahon, Kenichi Shihota,
Thien Le, Yana Narula

Objective: The objective of this project is to make utilization of the
HC-SR04 Ultrasonic Sensor,
Adafruit IO (The interface that connects a Raspberry Pi component to a
Dashboard), and a buzzer.
With the components, develop a script to measure the distances
that the Ultrasonic sensor captures and utilize this to measure the
distance,
alert based on thresholds and display a real-time distance in the UI itself.

"""

# Libraries
import RPi.GPIO as GPIO
import time
import digitalio
import board
from gpiozero import Buzzer

# import Adafruit IO REST client.
from Adafruit_IO import Client, Feed, RequestError

# Set to your Adafruit IO key.
# Remember, your key is a secret,
# so make sure not to publish it when you publish this code!
ADAFRUIT_IO_KEY = "aio_FEQN39S5rQB59DM0yhJ9DX8eNPI9"

# Set to your Adafruit IO username.
# (go to https://accounts.adafruit.com to find your username)
ADAFRUIT_IO_USERNAME = "team49805"
```

```
aio = Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)

try: # if we have a 'as3' and 'as3-but' feed
test = aio.feeds('as3')
test2 = aio.feeds('as3-but')
except RequestError: # create a digital feed
feed = Feed(name="as3")
feed2 = Feed(name="as3-but")
test = aio.create_feed(feed)
test2 = aio.create_feed(feed2)

# GPIO Mode (BOARD / BCM)
GPIO.setmode(GPIO.BCM)
# Set GPIO Pins (Connecting Hardware to Code)
GPIO_TRIGGER = 18
GPIO_ECHO = 24
# Set GPIO direction (IN / OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)
def distance(): # this function measures the distance of each iteration
based on the value given by the Ultrasonic Sensor
#Set Trigger to HIGH
GPIO.output(GPIO_TRIGGER, True)
#Set Trigger after 0.01ms to LOW
time.sleep(0.00001)
GPIO.output(GPIO_TRIGGER, False)

StartTime = time.time()
StopTime = time.time()
#Save StartTime
while GPIO.input(GPIO_ECHO) == 0:
StartTime = time.time()

#Save time of arrival
while GPIO.input(GPIO_ECHO) == 1:
StopTime = time.time()
#Time difference between start and arrival
TimeElapsed = StopTime - StartTime
#Multiply with the sonic speed (34300 cm/s) and divide by 2, because there
and back
distance = (TimeElapsed * 34300) / 2
return distance
# Main

if __name__ == '__main__':
```

```
# try and except to avoid error by CTRL+C in terminal
try:
# GPIO to Code for buzzer
buzzer = Buzzer(25)

while True:
# Measure the distance
dist = distance()
print("Measured Distance = %.1f cm" % dist)

# based on the button value in Adafruit IO (value in this case is data)
data = aio.receive(test2.key)

# if button off
if data.value == '0':
print("Buzzer Off")
elif data.value == '1':
# Threshold 1: if more than 40cm away buzz 3 times
if dist > 40:
print("Far 3 buzz")
buzzer.on()
time.sleep(0.5)
buzzer.off()
time.sleep(0.5)
buzzer.on()
time.sleep(0.5)
buzzer.off()
time.sleep(0.5)
buzzer.on()
time.sleep(0.5)
buzzer.off()
# Threshold 2: if more than 20cm away buzz 2 times
elif dist > 20:
print("2 buzz")
buzzer.on()
time.sleep(0.5)
buzzer.off()
time.sleep(0.5)
buzzer.on()
time.sleep(0.5)
buzzer.off()
# Threshold 3: if closer to less than 20cm buzz once
else:
print("Close 1 buzz")
buzzer.on()
```

```
time.sleep(2)
buzzer.off()
time.sleep(3)
dist = round(dist,1)
aio.send_data(test.key, str(dist) + "cm")
#Reset by pressing CTRL + C
except KeyboardInterrupt:
print("Measurement stopped by User")
GPIO.cleanup()
```

3. Testing & Feedback:

- Test the entire system to ensure distance measurements are accurately transmitted to Adafruit IO.
- Calibrate the system if necessary to improve measurement accuracy and alert responsiveness.

Deliverables:

- You MUST demonstrate your working IoT Distance Measurement and Alert System with Adafruit IO Integration setup to your instructor.
- Submit your project report before the deadline (no extensions except for reasons stated in the course outline, e.g., with a medical report).
- Your individual report must be entirely original, as its authenticity will be verified using standard tools for confirmation.

Chapter 2 - Project Results and Observations

Challenges:

- Overcoming challenges related to the limited documentation in the Adafruit module.
- Overcoming the Adafruit IO and Code Connection
- Debugging complexities in the IoT environment, require more time than traditional coding.
- Managing the electrical components and addressing issues in the hardware connections.
- Ensuring reliable readings from the Ultrasonic Sensor.

Enhancements for the Project:

- Implement an event listener to improve the responsiveness and smoothness of the Adafruit interface.
- Organize the hardware setup for better cable management and reliability.
- Improve documentation and resources for working with the Adafruit module to make it more accessible for future projects.

Chapter 3 - Conclusion

Moving Forward: In conclusion, the project achieved its objective by successfully getting data from the Ultrasonic sensor. The project provided valuable learning experiences in both hardware and software development within the IoT domain. Going forward, we aim to enhance the project's responsiveness and organization for future applications.

References:

- Raspberry Pi Documentation
- Anthony from 4980 IoT class.