CS 498 Internet of Things
Capstone project

Project Topic:

Pet Monitoring and Automatic Feeding System

Project Report

Group:

Li Yi (NetID: liyi2)

Pui Sze Pansy Ng (NetID: ppn2)

Table of Contents

1. Motivation

I love my dog. I think my dog loves me too. Due to the COVID-19 outbreak in the US, I am working from home these days. My dog likes to lay beside my chair and stay with me while I am working. But one day, I will be back to the office, leaving my dog himself at home. I always feel sorry for that. During the time I am out of home, no one feeds my dog, no one takes care of him. It concerns me very much. I would like to design something to monitor and to feed my dog when I am not at home. A pet monitor and automatic pet feeder with a camera would be a good option.

However, in the market, there are only some pet food dispensers with motion detection or by manual. These products are not good choices. Because sometimes raccoons, cats, birds, even rats visit my backyard. Those animals will trigger motion detection, then food will automatically be dispensed to the bowl. With pet food exposed, more and more wild animals will visit my backyard. I don't want to have a zoo in my backyard. I need a dispenser that can recognize my dog, and only dispenses pet food if a dog is detected. Luckily, we can leverage tensorflow object detection to recognize my dog. In this case, even raccoons, cats show up in front of the food dispenser, the food will not be dispensed as the camera does not recognize a dog in the frame.

Furthermore, when I am not at home, I would like to know when or how often my dog eats food. If my dog hasn't eaten for too long, there might be something wrong. I would like to get a notification or text message in this case, so that I can check out my dog by the backyard camera. On the automatic pet feeder, It can feed the pet by schedule or by a click. In addition, it sends a notification/text message to the phone everytime pets eat or pets haven't eaten for a long period of time. With the pet monitoring and auto feeding system, I am able to monitor my dog anytime,anywhere.

In summary, There are two problems that we need to solve in this project:

1. Dog detection - dispensing pet food only if a dog is detected in front of the food dispenser.
2. Pet monitoring - sending SMS or notifications to cell phones when the pet is having food or if the pet hasn't eaten for a long time.

2. Technical Approach and Implementation details

2.1 Tensorflow Object Detection

We have a Raspberry Pi that performs object detection. Picamera is connected to the raspberry. OpenCV-python and tensorflow are installed to perform object detection. After connected to Ardunio and distance sensor, executing python script -- Object_detection_picamera_May10.py
We would like to recognize dogs in the camera frame, but sometimes, my dog is detected as a dog, cat or teddy bear. so we added condition:
if ((int(classes[0][0]) == 17) or (int(classes[0][0] == 18) or (int(classes[0][0]) == 88))):
It means run the following code in the loop if only dog, cat, and teddy bear is recognized.

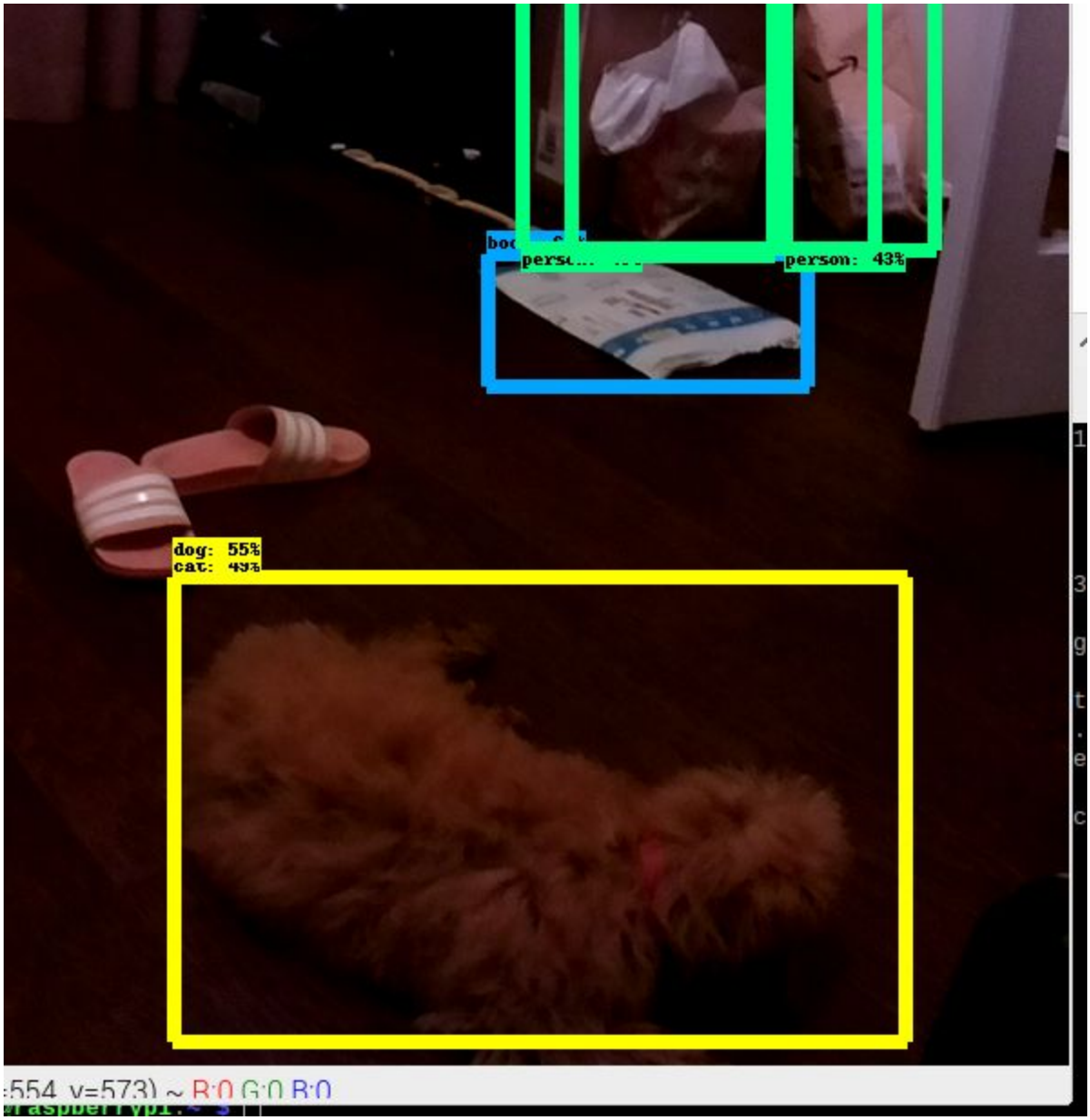

Figure 2.1.1 Screenshot of image of piCamera and Tensorflow object detection

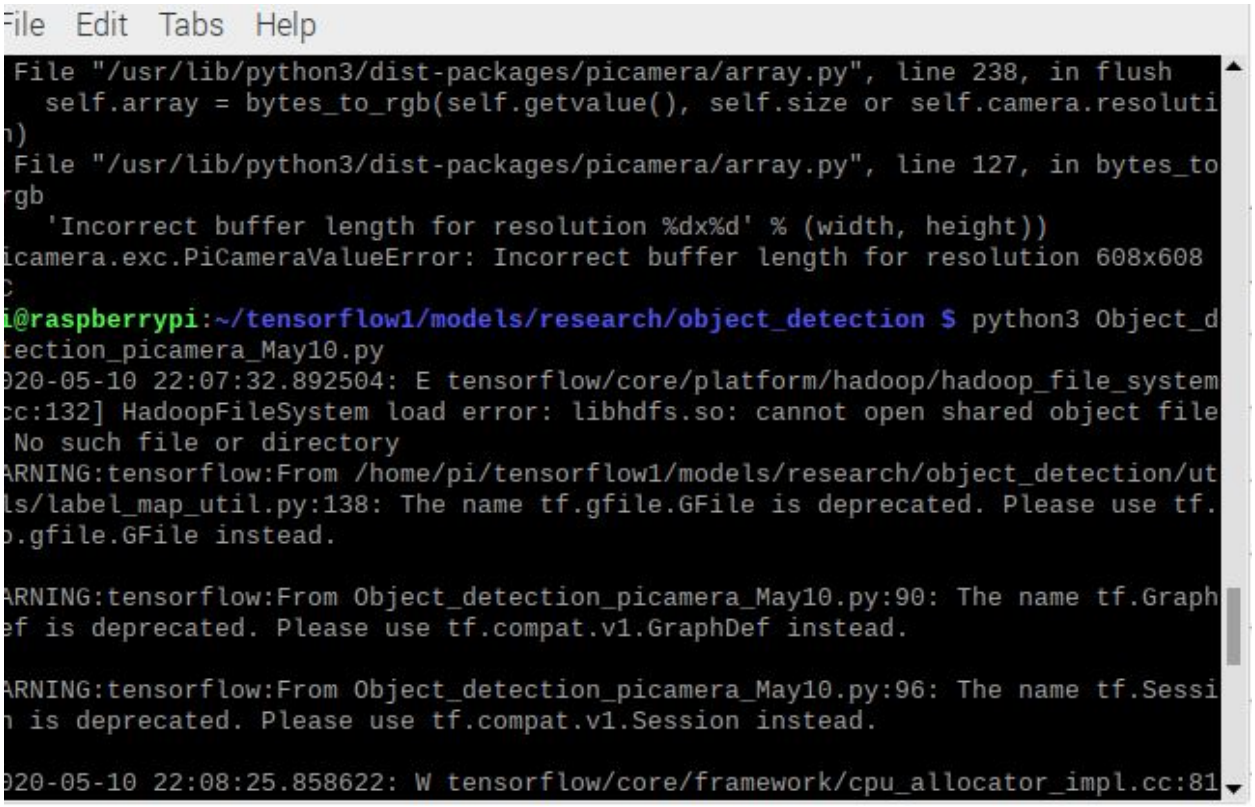

Figure 2.1.2 Screenshot of Raspberry pi terminal on object detection execution

2.2 Arduino Circuit Design

First, we have a distance sensor connected to the Arduino Uno board using PIN 7 and PIN 6. The distance calculation is:
Speed of sound v=340 m/s = 0.034 cm/us
Time = distance/speed t = s/v
Distance in cm:
S = t* 0.034/2

Second, we have a servo connected to PIN 3. The servo is driving the food dispenser to drop the dog food.
Third, the LED is connected to PIN 8. If the distance is smaller than 50 CM, and a dog is detected in the camera frame, LED light will be turned on.
Please see the following code for Arduino UNO board.

#include <SPI.h>
#include <Servo.h>

```
Servo myservo;
int val;
int trigPin = 7;
int echoPin = 6;
int ledPin = 8;
int distance; // centermeter (cm)
long duration;
int ledValue;
int receiveByte = 0;



void setup() {

  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  //pinMode(CAN0_INT, INPUT);
  pinMode(ledPin, OUTPUT);
  myservo.attach(3);
  myservo.write(0);


void loop() {

  ledValue = digitalRead(ledPin);
  // clearing trigpin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
  int receiveByte = 0
  if (Serial.available() > 0){
    receiveByte = Serial.read();
  }
  Serial.println(receiveByte);
  if (receiveByte != 0){
```

```
    if (distance <= 50) {
      digitalWrite(ledPin, HIGH);
      //myservo.attach(3);
      myservo.write(90);
      delay(5000);
      myservo.write(0);
      delay(20000);
      //myservo.detach();
    }
    else {
      digitalWrite(ledPin, LOW);
      myservo.write(0);
    }
  }
  Serial.println("");
  Serial.print(distance);
  Serial.println(" cm");
  Serial.print("LED= ");
  Serial.println(ledValue);
  delay(1000);   // send data per 1000ms
}
```

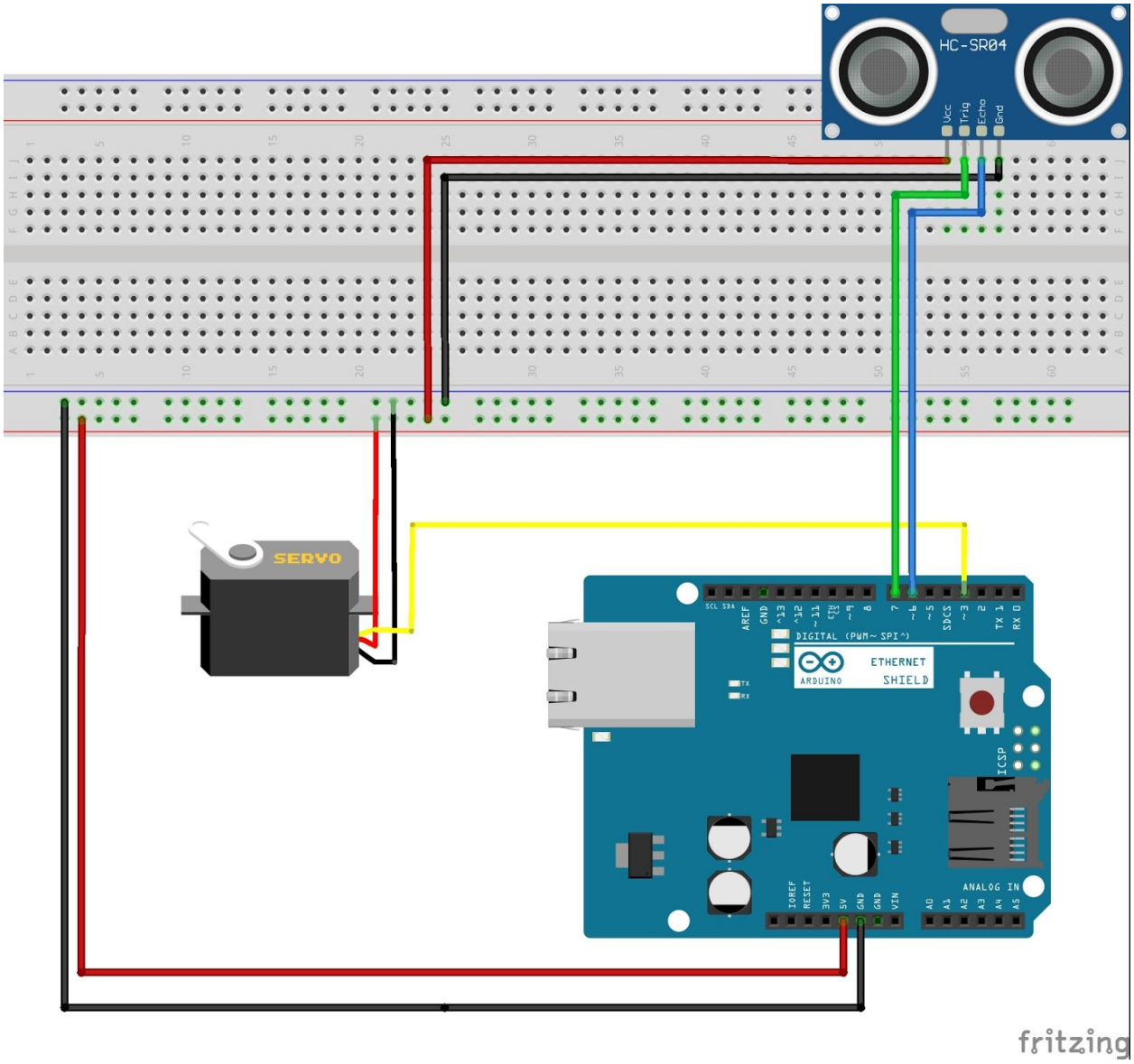

Figure 2.2.1: fritzing illustration

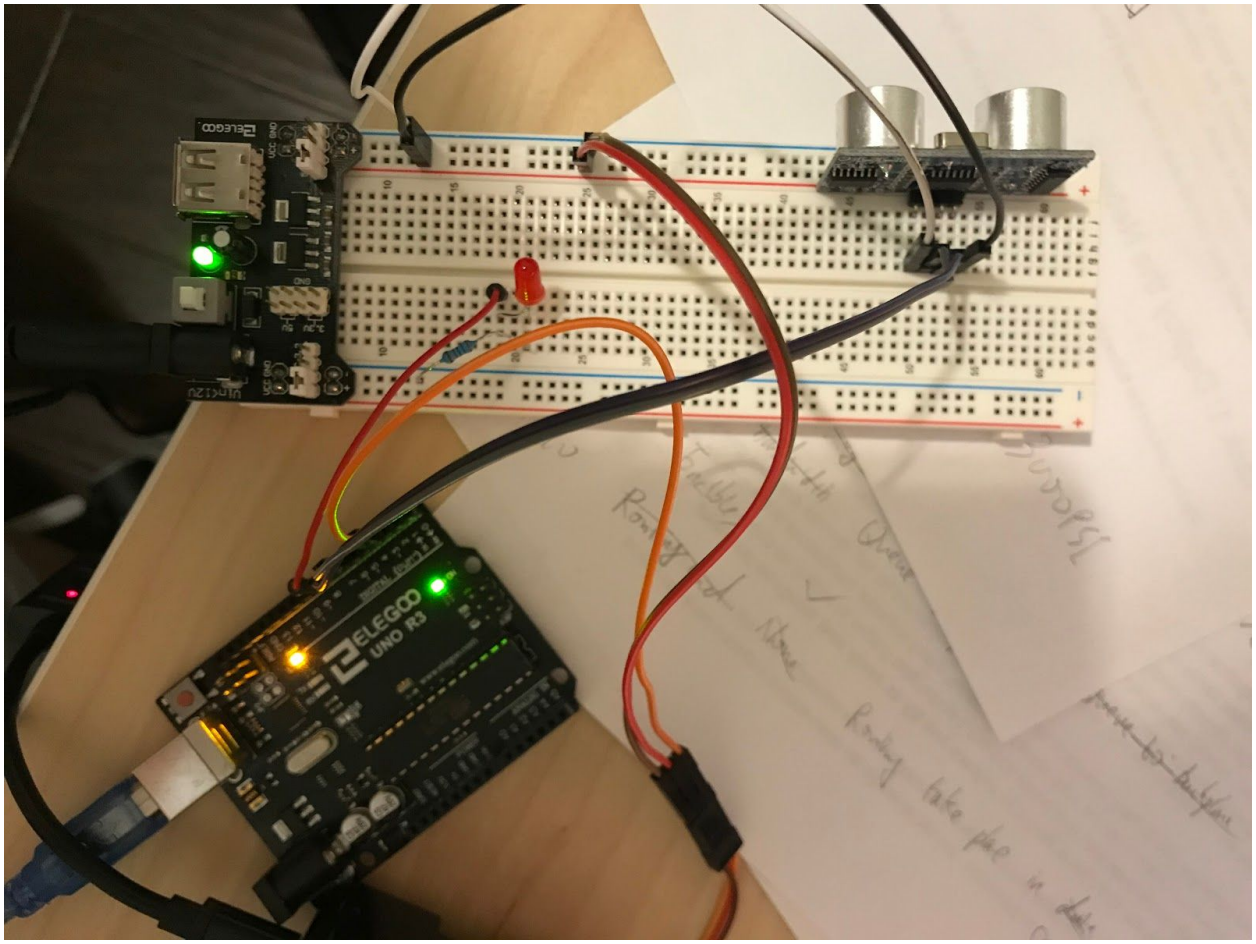

Figure 2.2.2: Image of the actual setup: Raspberry pi, Arduino Uno, and ultrasonic sensor

2.3 Feeding system

The feeding system is connected to a servo. If a dog is detected by tensorflow and distance is smaller than 50 cm, the servo connected to the Arduino UNO board will rotate the dispenser 90 degrees. The dog food then is dropped from the food container.

Figure 2.3.1 Image of actual setup: servo motor and food dispenser

2.4 Twilio SMS Notification

In order to send SMS messages when my dog is eating the food, we use Twilio as the message service. First, we need to create a Twilio account, and obtain account SID and token, then generate a message phone number. install Twilio CLI in Raspberry PI.

```
curl -o-
https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh |
bash
npm install twilio-cli -g
npm install twilio-cli@latest -g
```

After Twilio CLI is installed, create a message service on Raspberry Pi, and providing account SID and account token.

```
twilio api:messaging:v1:services:create --friendly-name "My first
Messaging Service"
twilio api:messaging:v1:services:phone-numbers:create --service-sid
MGXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX --phone-number-sid
PNXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Following the twilio example to send a message if a food dispenser is triggered.

```
from twilio.rest import Client
# Your Account Sid and Auth Token from twilio.com/console
# DANGER! This is insecure. See http://twil.io/secure
account_sid = 'ACXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
auth_token = 'your_auth_token'
client = Client(account_sid, auth_token)
message = client.messages \
    .create(
         body="Join Earth's mightiest heroes. Like Kevin Bacon.",
         messaging_service_sid='MGXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',
         to='+15558675310'
     )
print(message.sid)
```

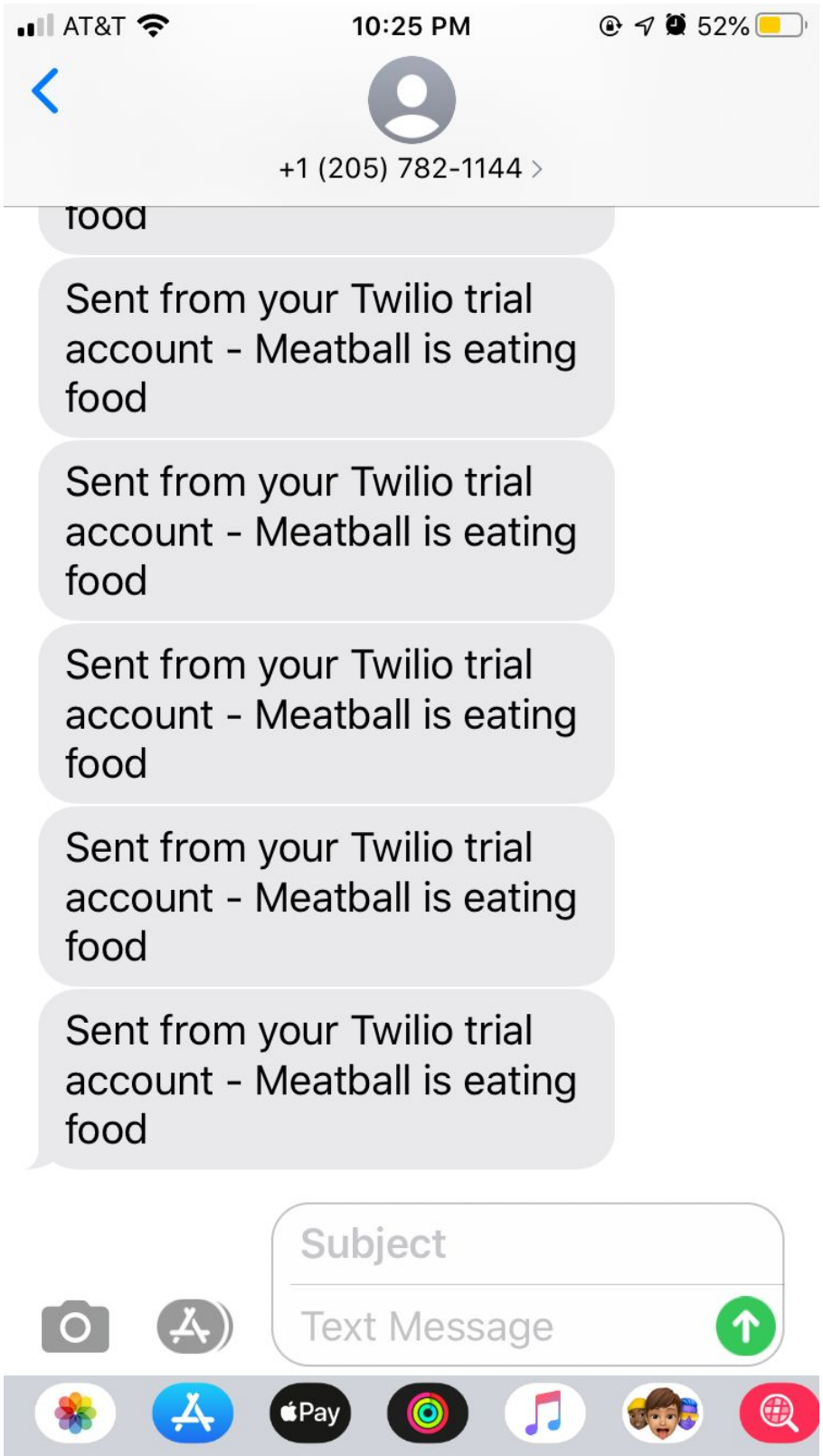

Figure 2.4.1: Screenshot of Twilio SMS messages

2.5 Integration with React Native and Google Dialogflow

To make it more like an actual product that can be used for deployment, we have integrated Twilio, AWS and Google Dialogflow. This product can also be integrated with Google Assistant and Alexa. React Native can provide a better visualization for this application This chatbot has three functions: 1. Change feeding time 2. Feed now 3. Enable SMS notifications when dog is detected.

First, we need to set up the chatbot logic in Google Dialogflow. Then, we need to figure out the intent - what kinds of action that we like to perform and entities - what are the substitute words for your actions or keywords. There is an option reply inside Dialogflow that we can reply with webhook fulfillment. We used Twilio, Python Flask, and ngrok to create a webhook and attach it as the reply. We also enable a Small Talk feature, which can handle random questions and responses such as who are you? How old are you?, in Dialogflow which can make this virtual assistant more humanized.

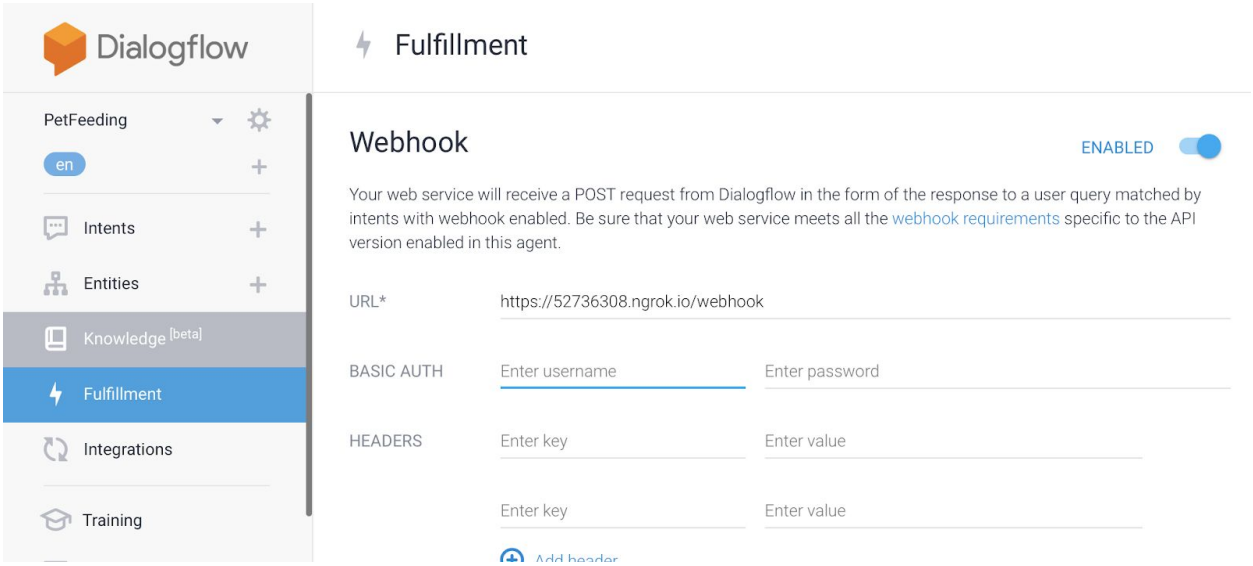

Figure 2.5.1: Screenshot of Dialogflow Webhook

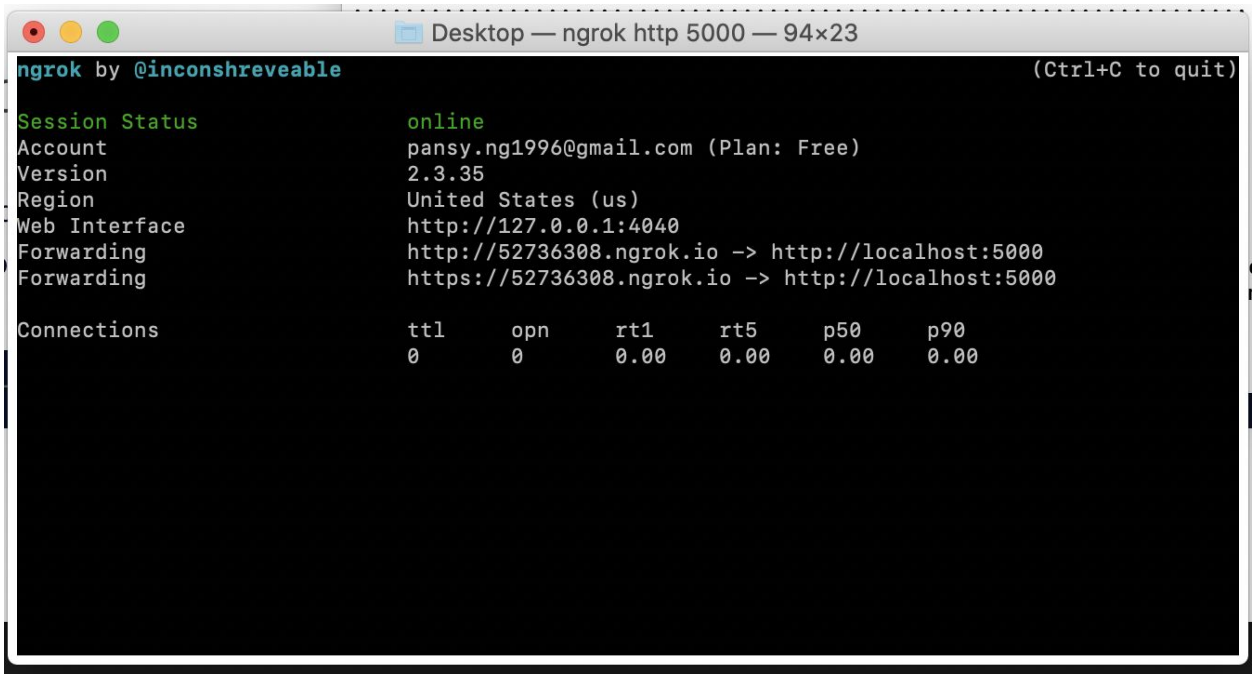

Figure 2.5.2: Screenshot of ngrok

Here is the sign in/ sign up page using AWS Cognito and Amplify withAuthenicator:

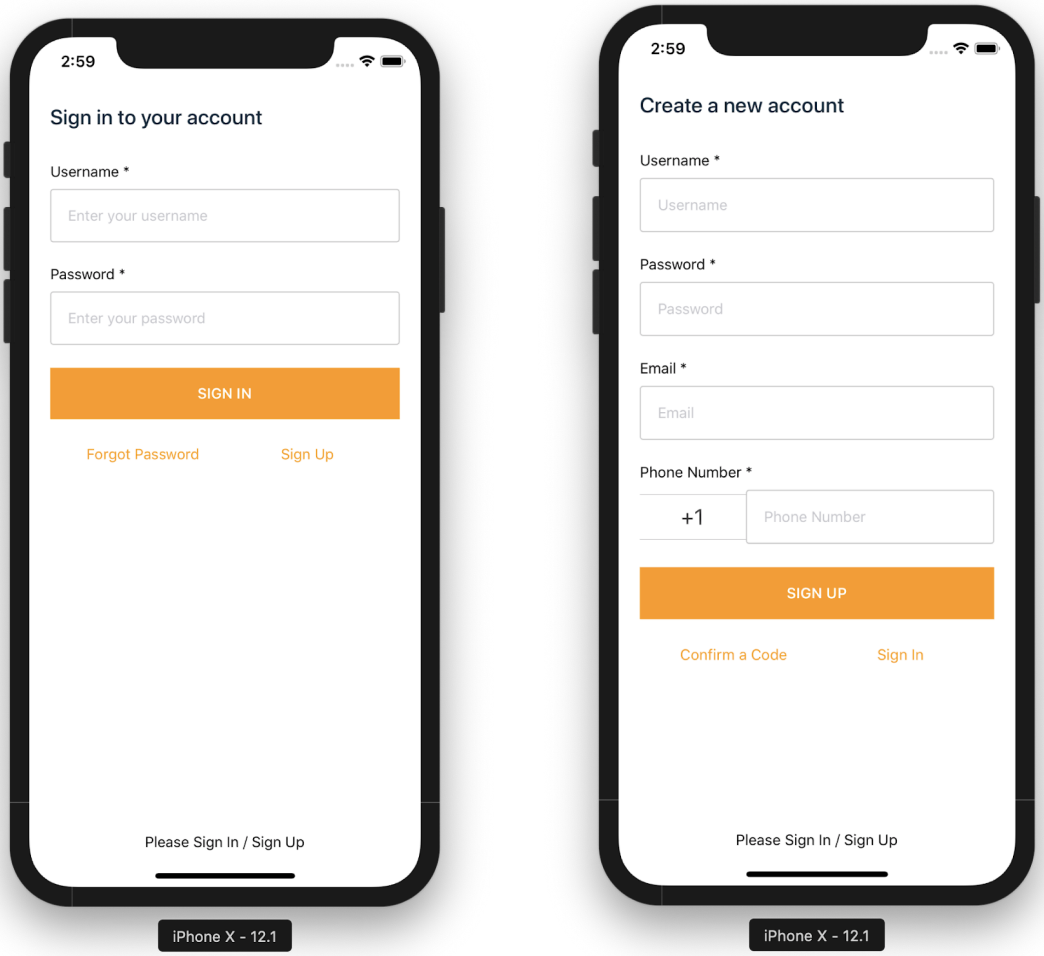

Figure 2.5.2: Screenshot of mobile application sign-in and sign-up page

Here is the chat application with our virtual smart dog feeder assistant meatball:

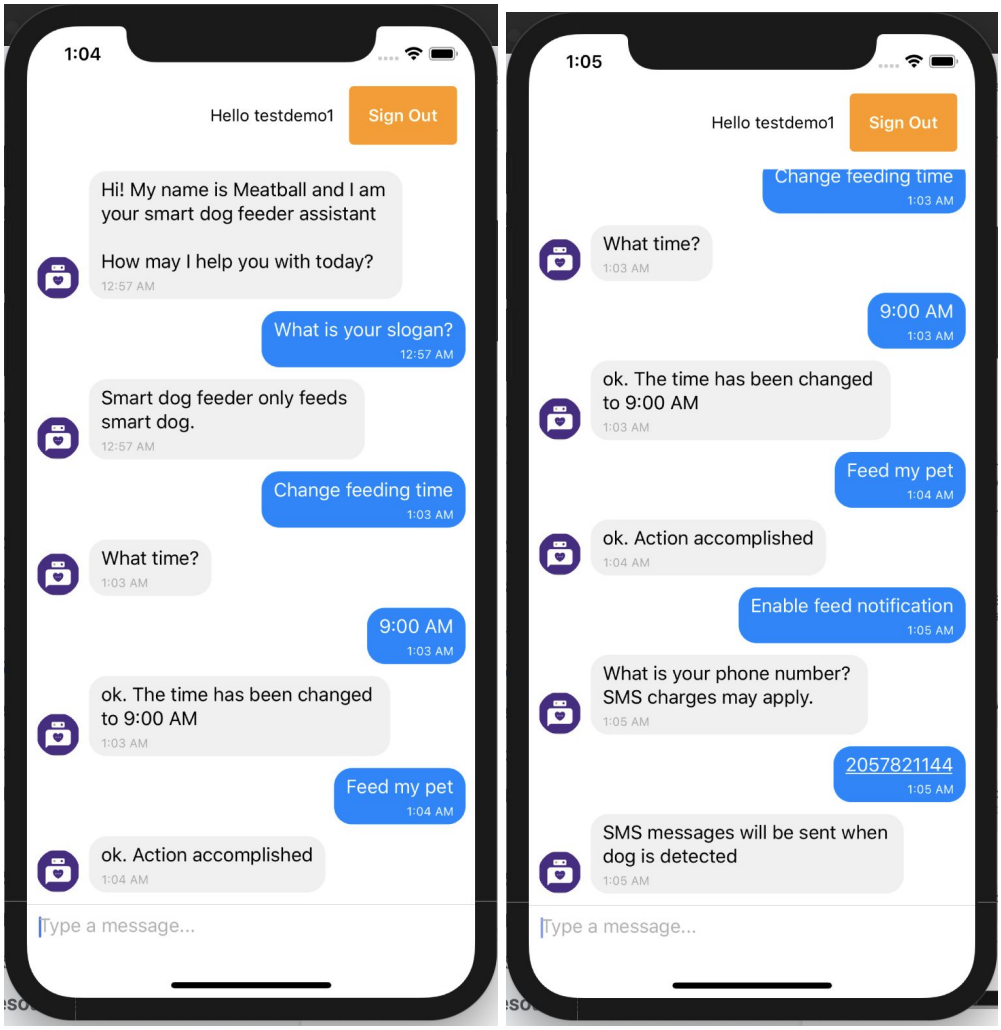

Figure 2.5.3: Screenshot of Mobile chatbot application

App.js > ...

```
// App.js

import React, { Component } from 'react';
import { StyleSheet, Text, View } from 'react-native';
import { GiftedChat } from 'react-native-gifted-chat';
import Amplify from 'aws-amplify';
import awsConfig from './aws-exports';
import { Dialogflow_V2 } from 'react-native-dialogflow';
import { dialogflowConfig } from './env';
import { withAuthenticator } from 'aws-amplify-react-native';

const BOT_USER = {
  _id: 2,
  name: 'PetFeeding',
  avatar: 'https://i.ibb.co/1KS99DK/Cocobot-icon-V201906.png'
};

Amplify.configure(awsConfig);
class App extends Component {
  state = {
    messages: [
      {
        _id: 1,
        text: `Hi! My name is Coco and I am your pet feeding assistant \n\nHow may I help you with today?`,
        createdAt: new Date(),
        user: BOT_USER
      }
    ]
  };

  componentDidMount() {
    Dialogflow_V2.setConfiguration(
      dialogflowConfig.client_email,
      dialogflowConfig.private_key,
      Dialogflow_V2.LANG_ENGLISH_US,
      dialogflowConfig.project_id
    );
  }

  handleGoogleResponse(result) {
```

App.js > ...

```
    let text = result.queryResult.fulfillmentMessages[0].text.text[0];
    this.sendBotResponse(text);
  }

  onSend(messages = []) {
    this.setState(previousState => ({
      messages: GiftedChat.append(previousState.messages, messages)
    }));

    let message = messages[0].text;
    Dialogflow_V2.requestQuery(
      message,
      result => this.handleGoogleResponse(result),
      error => console.log(error)
    );
  }

  sendBotResponse(text) {
    let msg = {
      _id: this.state.messages.length + 1,
      text,
      createdAt: new Date(),
      user: BOT_USER
    };

    this.setState(previousState => ({
      messages: GiftedChat.append(previousState.messages, [msg])
    }));
  }

  render() {
    return (
      <View style={{ flex: 1, backgroundColor: '#fff' }}>
        <GiftedChat
          messages={this.state.messages}
          onSend={messages => this.onSend(messages)}
          user={{
            _id: 1
          }}
        />
```

App.js > ...

```
      </View>
    );
  }
}

export default withAuthenticator(App, true);
```

Figure 2.5.4: Screenshot of the App.js

```
const awsConfig = {
    identityPoolId: 'xxxxx-xxxxxx-xxxx-xxxx-xxxxx',
    region: 'us-east-1',
    userPoolId: 'us-east-1_xxxxxxx',
    userPoolWebClientId: 'xxxxxxxxxxxxxxxx'
  }

  export default awsConfig;
```

Figure 2.5.5: screenshot of aws-export.js - For connection with AWS

```
// env.js

export const dialogflowConfig = {
    type: 'service_account',
    project_id: 'petfeeding-xxxxx',
    private_key_id: 'xxxxxxxx',
    private_key: '-----BEGIN PRIVATE KEY-----\nxxxxxxxxxxxxxxxxxxxxxxxxx5I1Lx5zRMFtyitEQ0kI/ntcgnEaAXKl2VuxLKIm5Yp6KS\nRTL3uxZsxGf6WOht
    client_email: 'dialogflow-xxxxxxpetfeeding-xxxxxx.iam.gserviceaccount.com',
    client_id: 'xxxxxxxxxxxxxxxxx',
    auth_uri: 'https://accounts.google.com/o/oauth2/auth',
    token_uri: 'https://oauth2.googleapis.com/token',
    auth_provider_x509_cert_url: 'https://www.googleapis.com/oauth2/v1/certs',
    client_x509_cert_url: 'https://www.googleapis.com/robot/v1/metadata/x509/dialogflow-bhccyu%40petfeeding-xxxxxxxxx.iam.gserviceaccou
  };
```

Figure 2.5.6: screenshot of env.js - For connection with Google dialogflow

3. Results

Video Demonstration

Here is our video demonstration:
Hardware side (Recorded by Li Yi) -
https://drive.google.com/open?id=1olHovTIIKdvF06wqwEZBLspSHdwPhBVG

Software side (Recorded by Pui Sze Pansy Ng) -
https://drive.google.com/open?id=1CjzhIDz9FYogJOxrvzcTU9YiSxQp0Xz6

Product performance

We have installed and tested the product indoor. At first, the dog needs to be trained to understand what to do to get the food dispensed. After the dog is adapted to the automatic food dispenser, he is willing to go to the eating place, sit in front of the camera, and wait for the food to be dropped to the bowl. Since there is no need to call or bark to ask for food, my dog now goes to his eating place whenever he is hungry.
Since we leveraged Tensorflow Object detection, only dogs are detected in the camera frame, food will be dispensed from the container. This is good for my dog especially when he is in the backyard. Sometimes wild animals such as racoons, squirrels, and opossums visit my backyard and take the dog food. Having this automatic dog food dispenser can avoid wild animals eating the dog food and furthermore avoid them visiting the backyard.
In addition to tensorflow object detection, we also added a distance ultrasonic sensor to detect the distance between the dog and the food bowl. When my dog is running around in the backyard, detected by tensorflow, The ultrasonic sensor prevents the food dispenser from being triggered. We configured the food is only dispensed if the dog is less than 30 cm to the food bowl.

Issues:

Although the automatic dog feeder works, there are still some issues that need to be resolved.

Servo is too small

We have a 9G SF0180 servo driving the food container which is a 500 ml water bottle. If the bottle is filled by the dog food, the servo cannot drive the heavy load. As a result, the food container can only be filled with 100g of dog food. This issue can be resolved by purchasing a larger servo, however, at COVID19 situation, purchasing a larger servo on Amazon takes too long to deliver.

Reliability

The automatic dog feeder is a system that has an Arduino Uno board, a Raspberry Pi, a servo and an ultrasonic sensor. All elements are connected by wires and power cables. It will be great to build a case encapsulating all parts to prevent dog chewing and swallowing. Ultimately, the automatic dog feeding system needs to be deployed in an outdoor environment. It needs to be waterproof, windproof and robust enough to take dropping damage.etc.

Port forwarding

Since we need to use ngrok for port forwarding, the computer needs to stay on for the whole time. Sometimes, bad internet connection will cause the chatbot to send error or null messages.

Sending Image has not been achieved

Since our project is called pet monitoring and automatic feeding system, we have achieved an automatic feeding system. Feed while the dog is detected. However, Monitoring has not been fully implemented. Sending images or streaming the video from the raspberry pi would be a great way to prove it is monitoring.

Takeaway:

This final project is a very interesting and useful project. We have combined what we have learnt from the previous lab, such as Raspberry pi with picamera on TensorFlow Lite Object Detection, Arudino, and ultrasonic sensor, and new technologies such as Servo motor, Twilio and Google Dialogflow.

4. Resources that we used

Twilio - Programmable SMS and Autopilot
https://www.twilio.com/

Google Dialogflow
https://dialogflow.com/

Webhook Tutorial
https://www.twilio.com/blog/build-whatsapp-chatbot-twilio-dialogflow-php

AWS Amplify
https://docs.amplify.aws/start/q/integration/react-native