BRAC UNIVERSITY

Inspiring Excellence

CSE461: Introduction to Robotics
Project Report
Project Title: Surveillance Bot

Group No: 04, CSE461 Lab Section: 07, Spring 2024	
ID	**Name**
23241062	Anindiya Majumder
21201428	Md. Mubashirul Islam
22101395	Tahmid Bin Haque
21201292	Nahiyan Ahamad Khan
21201241	Nirvik Shaha Rudra

Table of Contents

Introduction

In the realm of robotics, the amalgamation of hardware and software often heralds innovation, pushing the boundaries of what machines can achieve. Our project, titled "Surveillance Bot," epitomizes this fusion. With a Raspberry Pi at its core, this project aims to construct a versatile surveillance mechanism capable of detecting potential threats through image recognition, while also possessing the capability to respond with a buzzer for auditory alerts. As a group endeavor, we embarked on this journey to delve into the intricacies of robotics, combining theoretical knowledge with hands-on implementation to engineer a functional solution.

Project Features

The Surveillance Bot boasts an array of features that underscore its utility and effectiveness in surveillance scenarios. Primarily, it incorporates image recognition technology, enabling it to identify potential threats within its vicinity. Additionally, the integration of gas sensor and temperature and humidity sensor augments its surveillance capabilities, enabling it to detect suspicious activity. Upon detection, the bot activates its buzzer emitting audible alerts through a buzzer, alerting nearby personnel.

Components Required

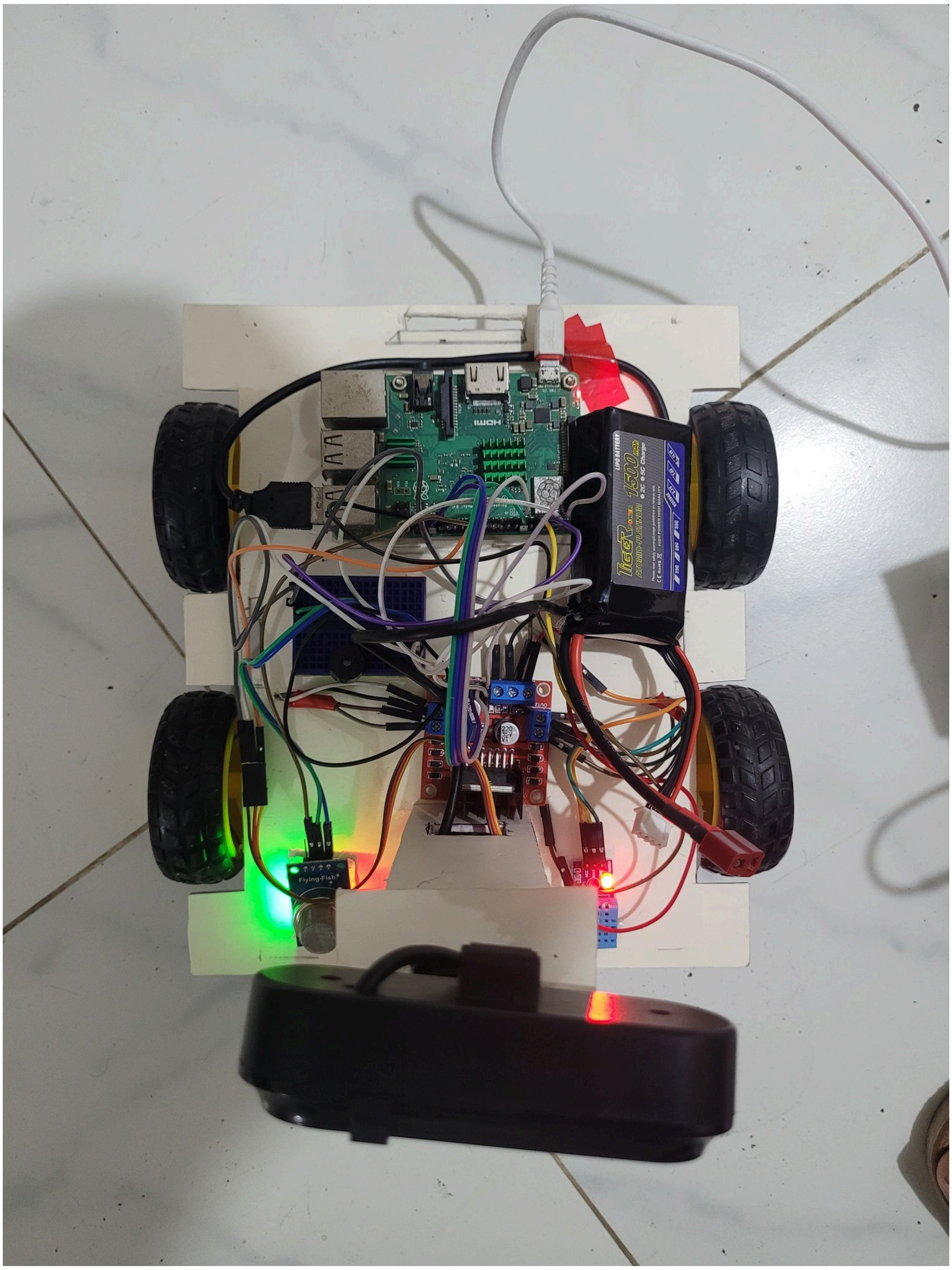
- Raspberry Pi: 3B+ model. The central processing unit is responsible for running the surveillance bot's software and coordinating its various components.
- Motor: 4 dc motors, for moving the robot.
- Wheels: 4 wheels attached to the motor.
- Motor Driver: L29810 to control the motors.
- Breadboard: For making connections between components.
- Webcam: A compatible webcam to capture visual data for image recognition purposes.
- Temperature and humidity sensor: DHT11 for temperature detection.
- Buzzer: An auditory alert system to notify nearby personnel upon detecting suspicious activity.
- Four-Wheeler Car System: A chassis with four wheels to provide mobility to the surveillance bot.
- Power Supply: Lipo 1500maH 11.1V battery.

- Circuitry and Wiring: Various electronic components and wiring to connect and integrate the different modules and sensors.
- Protective Enclosure: A protective casing to house the Raspberry Pi and other sensitive components, safeguarding them from external elements.
- Mounting Hardware: Jumper wires for circuit connections and other components for holding the sensors and other devices.
- Networking Components: Build in Pi wifi, HTTP, SSH, VNC.
- Development Tools: Raspberry Pi OS Buster version, Thony, Open CV, Flask, Bash.

Methodology

1) Connected a temperature and humidity sensor and a gas sensor for the Raspberry Pi.
2) Ran open CV on the Pi and then ran Haaracascade frontal face detection algorithm for face detection and made bounding boxes for marking the face.
3) Sending the video through Flask server and HTTPs request so that we can access the video anywhere by requesting the server.
4) Connected a servo with the Pi and attached to the camera for a different angle view.
5) Connected a buzzer alarm system so that it can make a sound after detecting the face.
6) Connected four DC motors with motor driver and gave the motor 12V, to move the robot.
7) Made the chassis for the robot with the PVC board for protection.

System Architecture



Implementation Details

1) Controller unit: We are using a Raspberry Pi 3b+ for our controller. It has 1GB LPDDR2 SDRAM, (ARMv8) 64-bit SoC @ 1.4GHz processor and

32gb storage which will be provided by a sdcard. We are running the legacy version of Raspberry Pi Buster OS on it.

2) Sensors: Taking inputs and calibrating them for our calculations. We are taking inputs from three different sensors. One is a camera, a smoke detector and a temperature & humidity sensor. DHT11 temperature sensor and MQ2 sensors are a passive sensor which needs power. We provided a common VCC and ground which was taken from the PIS GPIO pin to power all the sensors. The output GPIO pins were pin 7 for the DHT11 sensor and its 4 for the MQ2 sensor. Calculating the temperature and humidity is done by the Adafruit Library's DHT method. Also if the MQ2's threshold is going above a certain threshold we can say that a good amount of gass was detected. Finally, we have a web camera which gives input to the pi using USB A. We take an image 24 times a second to get a 24fps continuous video stream.

3) Actuators: We have two DC motors and one servo motor as our actuators. To move our robot we are taking input from the keyboard and binding them with an arrow key. The dc motors are connected to a motor driver which is also connected to a lipo battery. The input of the drivers are taken from Raspberry Pi 13. 16, 17, 18 pins. The servo is connected to our breadboard's common vcc and ground for power and the input for PWM is connected to pis 11 pins.

4) Power source: We are using two power sources here. One is a lipo battery with 1500mah capacity which is to drive the motors and another one is a power bank of 20000mah to power the Raspberry Pi.

5) Software Libraries and Algorithms: We are taking input from the webcam and running facial detection algorithm on it. It is done by pythons opencv module. A custom model is also used here to get the best output of the pi.

6) Communication: All of the communication to the pi is being done by wifi. The pi and the base station(another pc that isn't pi) are connected to the same network. We are using ssh and vnc to send commands, run scripts and monitor the activities of the robot. Also, we set up a http flask server on the Raspberry Pi to host the captured video of the webcam.

Results and Analysis:

After completing the project, we tested it multiple times and checked the outcome. It worked successfully. The camera is working and giving 24 fps data nicely, servo is working fine. All the sensors and software are working and correctly giving the data. Motor driving speed was Enough to move the robot.

Future Enhancements

We added so many options, but we can make it better and have a plan for this. First of all, the chassis of the bot is made of plywood. We will make it strong with nonheavy metal. Then, we can improve our algorithm for the face detection. The control of the motor of movement can be made autonomous by running a new algorithm which is our plan to work with in future. Last, but not least we have a plan to add effective sensor data and use that efficiently to enhance the practicality.

Ethical Considerations

We did not use any harmful device or system which would hurt someone. As it is detecting someone's face, it is just alerting with the sound of the buzzer.

Source Code Repository

```
#webstream.py for streaming

import cv2
import numpy
from flask import Flask, render_template, Response, stream_with_context, request

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

video = cv2.VideoCapture(0)
video.set(cv2.CAP_PROP_FRAME_WIDTH, 426)
video.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)

app = Flask('__name__')

def video_stream():
   while True:
      ret, frame = video.read()

      #frame = cv2.resize(frame, (320, 360))

      # Convert to grayscale
      gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

      # Detect the faces
      faces = face_cascade.detectMultiScale(gray, 1.1, 4)
      if type(faces) == numpy.ndarray:
         print("Enemy Detected!!")
      else:
         print("No enemy found")
      # Draw the rectangle around each face
      for (x, y, w, h) in faces:
         cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
      if not ret:
         break;
      else:
         ret, buffer = cv2.imencode('.jpeg',frame)
         frame = buffer.tobytes()
         yield (b' --frame\r\n' b'Content-type: imgae/jpeg\r\n\r\n' + frame +b'\r\n')

@app.route('/camera')
def camera():
   return render_template('camera.html')

@app.route('/video_feed')
def video_feed():
   return Response(video_stream(), mimetype='multipart/x-mixed-replace; boundary=frame')

app.run(host='0.0.0.0', port='5000', debug=False)




#Camera_servo.py for servo control
```

```
import RPi.GPIO as GPIO
import keyboard
from time import sleep
GPIO.setmode(GPIO.BCM)
GPIO.setup(11, GPIO.OUT)
pwm=GPIO.PWM(11, 50)
pwm.start(0)

while True:
   if keyboard.read_key()=="f":
      pwm.ChangeDutyCycle(2.5) #push the motor to 0 degree

   elif keyboard.read_key()=="g":
      pwm.ChangeDutyCycle(7.25)

   elif keyboard.read_key()=="h":
      pwm.ChangeDutyCycle(12)

pwm.stop()
GPIO.cleanup()
```

```
#move.py for moving the rover in forward, backward, right and left directions
from gpiozero import Motor
import curses

#OUT4 => 18
#OUT3 => 13
#OUT2 => 16
#OUT1 => 17

flmotor = Motor(forward=16, backward=17)
frmotor = Motor(forward=18, backward=13)
#blmotor = Motor(forward=9, backward=11)
#brmotor = Motor(forward=10, backward=12)

def left():
#   print('Left ...')
   flmotor.stop()
   frmotor.forward()
#   blmotor.backward()
#   brmotor.forward()
```

```
def right():
#   print('Right ...')
    flmotor.forward()
    frmotor.stop()
#   blmotor.forward()
#   brmotor.backward()

def forward():
#   print('Forwarding ...')
    flmotor.forward()
    frmotor.forward()
#   blmotor.forward()
#   brmotor.forward()

def reverse():
#   print('Reversing ...')
    flmotor.backward()
    frmotor.backward()
#   blmotor.backward()
#   brmotor.backward()

def stop():
#   print('Stopping ...')
    flmotor.stop()
    frmotor.stop()
#   blmotor.stop()
#   brmotor.stop()

actions = {
    curses.KEY_UP:    forward,
    curses.KEY_DOWN:  reverse,
    curses.KEY_LEFT:  left,
    curses.KEY_RIGHT: right,
}

def main(window):
    next_key = None
    while True:
        #curses.halfdelay(1)
        if next_key is None:
            key = window.getch()
        else:
            key = next_key
            next_key = None
        if key != -1:
            # KEY PRESSED
            #curses.halfdelay(3)
            action = actions.get(key)
            if action is not None:
                action()
            next_key = key
            while next_key == key:
                next_key = window.getch()
            if key not in [curses.KEY_UP, curses.KEY_DOWN, curses.KEY_LEFT,
curses.KEY_RIGHT]:
                stop()
            # KEY RELEASED
            stop()

curses.wrapper(main)
```

sensors.py for sensor interfacing

```
import Adafruit_DHT as dht11
import RPi.GPIO as GPIO
import time

gas_pin = 7
temp_pin = 4

GPIO.setmode(GPIO.BCM)
GPIO.setup(gas_pin, GPIO.IN)

temp = dht11.DHT11

while True:
    hum, tem = dht11.read_retry(temp, temp_pin)
    gas = GPIO.input(gas_pin)

    print("...................")

    if hum != None and tem != None:
        print(f"Temp: {tem}*C Humidity: {hum}%")
    else:
        print("Failed to get temperature data")

    if gas == GPIO.LOW:
        print("No excessive gas found")
    else:
        print("Excessive gas found")

    time.sleep(1)
```

Conclusion

In conclusion, the development of the Surveillance Bot has been an enriching experience, encapsulating the principles and applications of robotics in a tangible solution. Through meticulous planning, collaborative effort, and iterative refinement, we have successfully engineered a surveillance mechanism that showcases the convergence of hardware and software technologies. Moving forward, we envision further enhancements and optimizations to augment the bot's capabilities and applicability in diverse real-world scenarios.

Video Presentation

https://www.youtube.com/watch?v=vp3VAwmdAKc

References

1) https://www.raspberrypi.com/documentation/