Introduction

Statement of Purpose

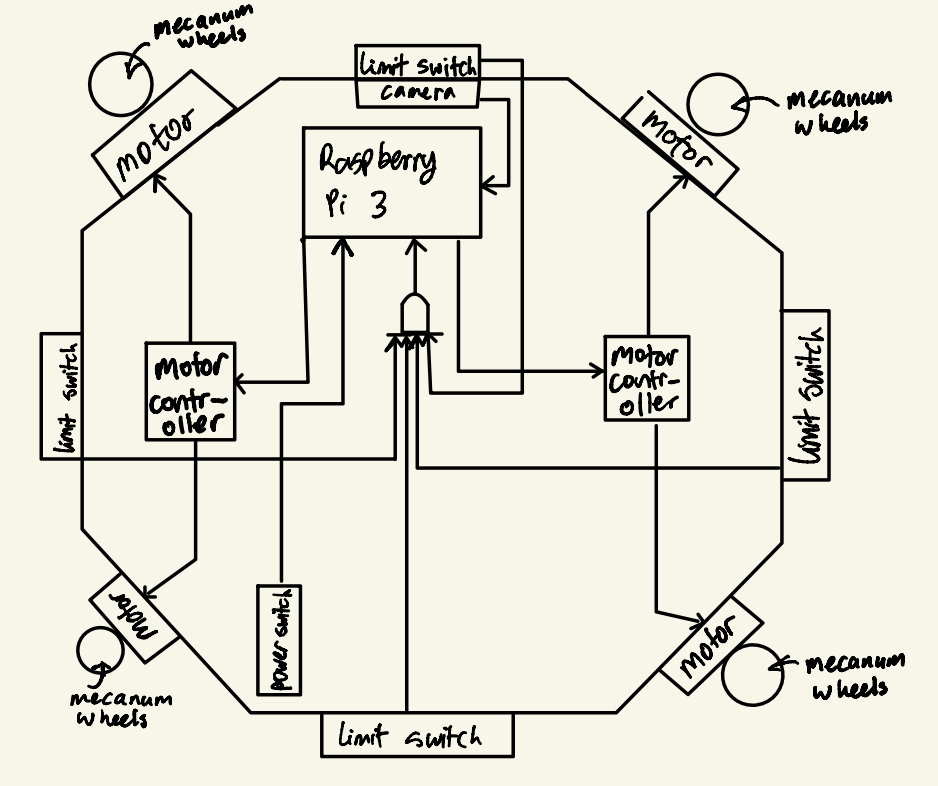
Many college students are carrying too many things around with them. The team’s solution to this problem is to create an autonomous car that follows a student and can carry a small bit of their load. Using a camera and limit switches, the autonomous robocar can detect its surroundings and move accordingly. The goal of this project is to free up students’ hands by providing them with a mindless solution that follows them wherever they go. This car will use a variety of cameras and sensors to follow people and avoid obstacles while carrying an object that fits within a 1’ x 1’ x 1’ space.

Features and Benefits

One of the main features of this project is that it uses computer vision to follow a person by using their relative position. The autonomous car also uses mecanum wheels to navigate around, so it can move in all 360 degrees of motion while keeping the camera in front facing the target. It also has limit switches that can sense if the car runs into obstacles, and will stop the car for five seconds to let the user clear the car’s path or move it accordingly.

Design

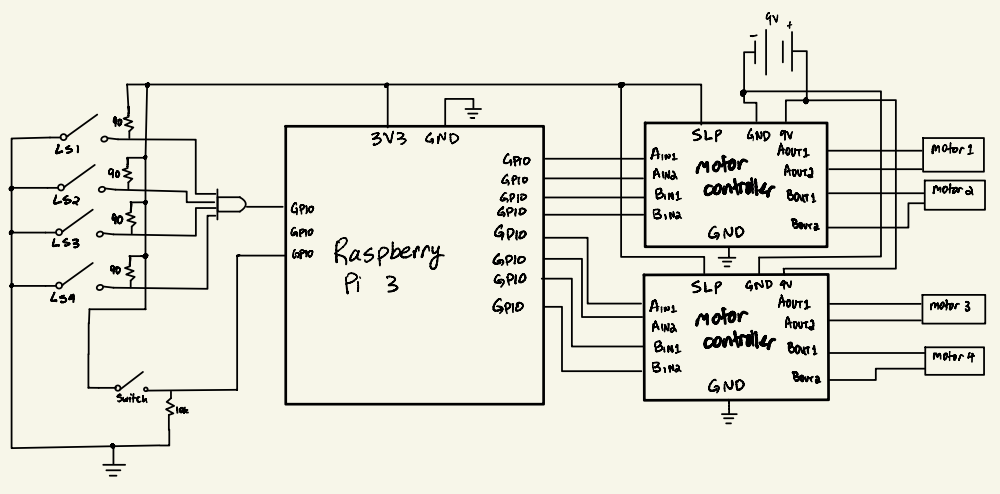
System Overview

****

**Figure 1: Block Diagram of Autonomous Car Carrier**

Figure 1 is in the shape of the chassis of the car. The central computer being used is a Raspberry Pi 3, which takes in the limit switches, camera, power switch, and reset button as inputs. The Raspberry Pi then sends signals to the motor controllers and the motors, which function as outputs. The four limit switches feed into a quadruple input AND gate and send the single signal to the Raspberry Pi 3. The camera is constantly sending images to the Raspberry Pi 3 to process, the power switch turns the motors on and off.

Design Details



**Figure 2: Circuit Schematics of Car**

Figure 2 shows the schematics and electrical components used to build the car. All of the limit switches are pulled up so that if any of them are triggered, the AND gate will send a 0 to the Raspberry Pi 3. Meanwhile, the power switch is connected to a pulldown resistor. For the outputs of the Raspberry Pi, all of the motor controllers are connected straight to the Raspberry Pi and also connected to a 9V battery to provide the motors with power.

The formula used to calculate the value of the pullup and pulldown resistors was:

*RMAX =*

The provided voltage is 3.3V running at 20 mA, and the required voltage for the Raspberry Pi to read a digital high is 1.8 V, the pullup resistor’s maximum value came out to 90 ohms.

The flow diagram (see Appendix A) for the code (see Appendix B) in the Raspberry Pi shows the logic that decides where the robot will go next by comparing the new x-position and size to the old x-position and size. The robot then moves accordingly, and gets a new image from the camera (see Appendix C), and if the limit switch is triggered, the code sleeps for five seconds.

Results

After twelve weeks of working on this project, a very early and simple prototype was designed. The team designed a chassis and was able to connect all the parts and motors together to create a working robocar with a power switch and limit switches. A demo code was also developed that can control the robocar, the motors, and the read the limit switches, and it uses a random number generator to simulate receiving images from the camera. An object detection module was also coded that can detect any people in the camera’s vision and returns the bounds box back to the computer to update the x-position and size, so the robocar can move accordingly. Overall, the results were positive, and with more computing power, a full working demo could be made, but even in the last twelve weeks, the chassis was built, the code for the motors was finalized, and the code for the camera was also developed, which were the three main components and goals of the project.

Problems and Challenges

One of the major problems the team faced during designing this project was that the computer vision libraries couldn’t be built on the Raspberry Pi 3, so the computer vision could only be tested on a laptop. The team considered many options such as installing OS with OpenCV preinstalled or building the library from source. None of these options worked with the Raspberry Pi, and after some more research, it became apparent that the Raspberry Pi 3 doesn’t have enough computing power to build the libraries, which is why the testing and demos for the computer vision and object detection were done on a separate laptop with enough computing power. Another problem the team faced was controlling the motors and reading the signals from the switches simultaneously. Initially, both the RPi and gpiozero libraries were being used for the motors and switches, but the team learned that the Raspberry Pi can’t handle two different GPIO modes at once, so the team switched from RPi to gpiozero libraries for every component, and then the code and parts were able to function properly.

Future Plans

The future plans for this project would be to get a faster and more powerful computer that can run the computer vision and object detection modules smoothly. Along with that, designing a box to hold the textbooks and school supplies would be a top priority. Designing a PCB to house all the connections and parts instead of using wires would also be in consideration if the team wanted to explore that option, but it also isn’t as much of a priority as getting the camera integrated with motors and box connected to the chassis.

References  
[1]*Projects.raspberrypi.org*, 2021. [Online]. Available: https://projects.raspberrypi.org/en/projects/build-a-buggy/2. [Accessed: 11- Dec- 2021].

[2]S. Monk, "Computer vision with the Raspberry Pi", *O’Reilly Media*, 2021. [Online]. Available: https://www.oreilly.com/content/raspberry-pi-cookbook-computer-vision/. [Accessed: 11- Dec- 2021].

[3]*Projects.raspberrypi.org*, 2021. [Online]. Available: https://projects.raspberrypi.org/en/projects/physical-computing/14. [Accessed: 11- Dec- 2021].

[4]*Projects.raspberrypi.org*, 2021. [Online]. Available: https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/2. [Accessed: 11- Dec- 2021].

[5]"Adafruit DRV8833 DC/Stepper Motor Driver Breakout Board", *Adafruit Learning System*, 2021. [Online]. Available: https://learn.adafruit.com/adafruit-drv8833-dc-stepper-motor-driver-breakout-board?view=all. [Accessed: 11- Dec- 2021].

[6]"Python Programming Tutorial: Getting Started with the Raspberry Pi - learn.sparkfun.com", *Learn.sparkfun.com*, 2021. [Online]. Available: https://learn.sparkfun.com/tutorials/python-programming-tutorial-getting-started-with-the-raspberry-pi/experiment-1-digital-input-and-output. [Accessed: 11- Dec- 2021].

[7]2021. [Online]. Available: https://www.electronics-tutorials.ws/logic/pull-up-resistor.html#:~:text=A%20pull%2Dup%20resistor%20connects,keep%20the%20given%20input%20LOW. [Accessed: 11- Dec- 2021].

[8]"FAQ on Object Detection Using SSD mobilenet!", *Medium*, 2021. [Online]. Available: https://madhumitamenon.medium.com/faq-on-object-detection-using-ssd-mobilenet-b8bf31924601. [Accessed: 11- Dec- 2021].

[9]*Youtube.com*, 2021. [Online]. Available: https://www.youtube.com/watch?v=Vg9rrOFmwHo. [Accessed: 11- Dec- 2021].

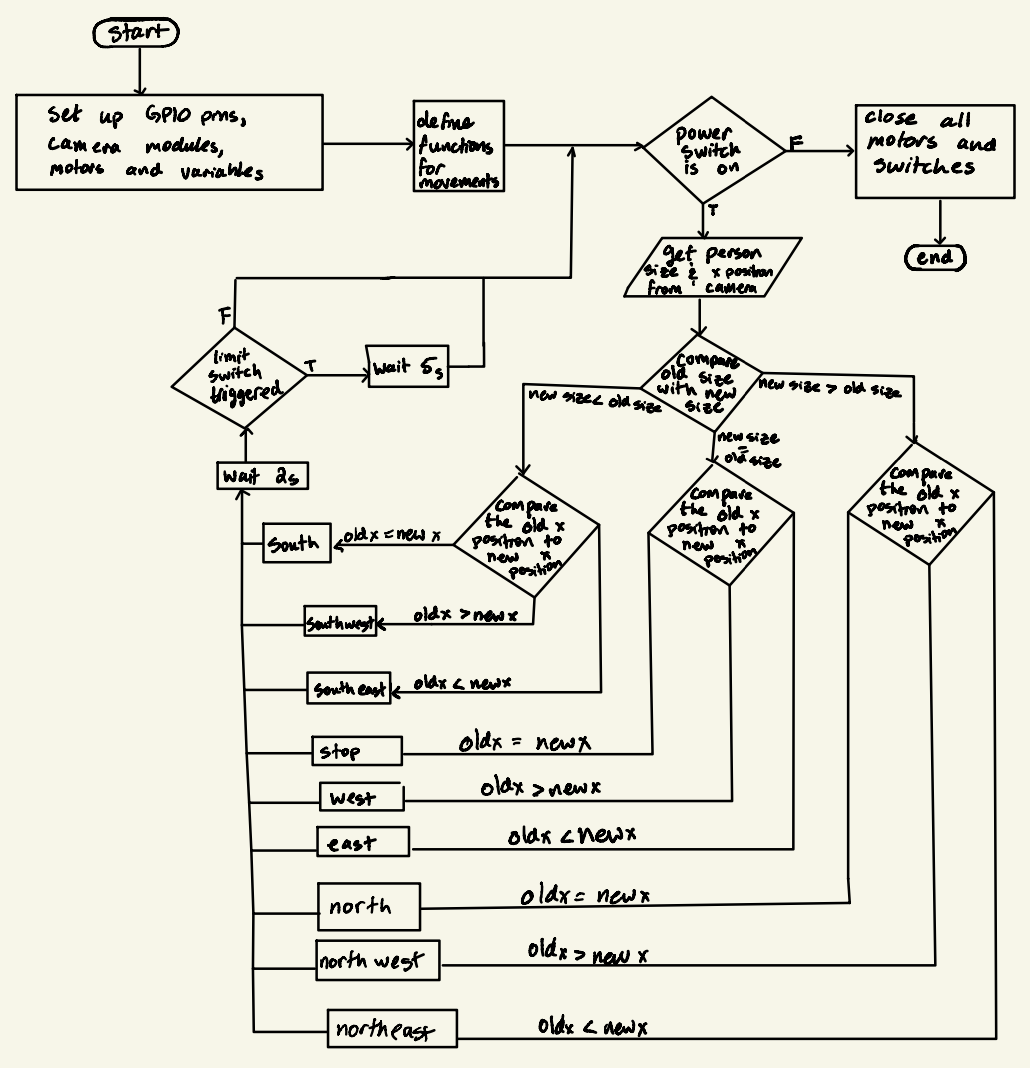
[10]A. Rosebrock, "Install OpenCV 4 on your Raspberry Pi - PyImageSearch", *PyImageSearch*, 2021. [Online]. Available: https://www.pyimagesearch.com/2018/09/26/install-opencv-4-on-your-raspberry-pi/. [Accessed: 11- Dec- 2021].

[11]*Youtube.com*, 2021. [Online]. Available: https://www.youtube.com/watch?v=rdBTLOx0gi4. [Accessed: 11- Dec- 2021].

[12]"GitHub - djmv/MobilNet\_SSD\_opencv: MobilNet-SSD object detection in opencv 3.4.1", *GitHub*, 2021. [Online]. Available: https://github.com/djmv/MobilNet\_SSD\_opencv. [Accessed: 11- Dec- 2021].

Appendices

Appendix A



**Figure 3: Flowchart for code inside Raspberry Pi 3**

Appendix B

| from random import randint, seed from gpiozero import Motor, Button import time  old\_x = 5 old\_size = 5 new\_x = 5 new\_size = 5 seed(1) front = None side = None  motor1 = Motor(25, 22) motor2 = Motor(24, 23) motor3 = Motor(10, 9) motor4 = Motor(16, 26) limit\_switch = Button(17) power\_switch = Button(4)   def north():  motor1.forward()  motor2.forward()  motor3.forward()  motor4.forward()  print("north")   def northeast():  motor1.stop()  motor2.forward()  motor3.forward()  motor4.stop()  print("northeast")   def east():  motor1.backward()  motor2.forward()  motor3.forward()  motor4.backward()  print("east")   def southeast():  motor1.backward()  motor2.stop()  motor3.stop()  motor4.backward()  print("southeast")   def south():  motor1.backward()  motor2.backward()  motor3.backward()  motor4.backward()  print("south")   def southwest():  motor1.stop()  motor2.backward()  motor3.backward()  motor4.stop()  print("southwest")   def west():  motor1.forward()  motor2.backward()  motor3.backward()  motor4.forward()  print("west")   def northwest():  motor1.forward()  motor2.stop()  motor3.stop()  motor4.forward()  print("northwest")   def stop():  motor1.stop()  motor2.stop()  motor3.stop()  motor4.stop()  print("stop")   while True:  while power\_switch.value == 1:  new\_x = randint(0, 10)  new\_size = randint(0, 10)  print("Old x: " + str(old\_x) + "|New x: " + str(new\_x))  print("Old size: " + str(old\_size) + "|New size: " + str(new\_size))   front = new\_size - old\_size  side = new\_x - old\_x  print("Front: " + str(front) + " |Side: " + str(side))   if limit\_switch.value:  print("triggered")  stop()  time.sleep(5)  print("pause over")  continue   if front > 0:  if side > 0:  northeast()  elif side < 0:  northwest()  else:  north()  elif front < 0:  if side > 0:  southeast()  elif side < 0:  southwest()  else:  south()  else:  if side > 0:  east()  elif side < 0:  west()  else:  stop()   old\_x = new\_x  old\_size = new\_size  time.sleep(2)  break motor4.close() motor3.close() motor2.close() motor1.close() limit\_switch.close() power\_switch.close() |
| --- |

**Figure 4: Code for controlling motors and reading limit switches and power switches**

Appendix C

| import cv2 #img = cv2.imread('cc.jpg') #thres = 0.5  classNames=[] classFile = '/home/pi/Desktop/ECE120 HL/ObjectDetection/coco.names' with open(classFile,'rt') as f:  classNames = f.read().rstrip('\n').split('\n')  print(classNames)  configPath = '/home/pi/Desktop/ECE120 HL/ObjectDetection/ssd\_mobilenet\_v3\_large\_coco\_2020\_01\_14.pbtxt' weightsPath = '/home/pi/Desktop/ECE120 HL/ObjectDetection/frozen\_inference\_graph.pb'  net = cv2.dnn\_DetectionModel(weightsPath,configPath) net.setInputSize(320,320) net.setInputScale(1.0/127.5) net.setInputMean((127.5,127.5,127.5)) net.setInputSwapRB(True)   def getObjects(img,thres,nms,draw=True,objects=[]):  classIds, confs, bbox = net.detect(img, confThreshold=thres,nmsThreshold=nms)  #print(classIds, bbox)  if len(objects) == 0:  objects=classNames  objectInfo=[]  if len(classIds) != 0:  for classId, confidence, box in zip(classIds.flatten(),confs.flatten(),bbox):  className = classNames[classId - 1]  if className in objects:  objectInfo.append([box, className])  if(draw):  cv2.rectangle(img,box,color=(0,255,0),thickness=2)  cv2.putText(img,classNames[classId-1].upper(),(box[0]+10,box[1]+30),cv2.FONT\_HERSHEY\_COMPLEX,1,(0,255,0),2)  cv2.putText(img,str(round(confidence\*100,2)), (box[0] + 150, box[1] + 30), cv2.FONT\_HERSHEY\_COMPLEX, 1,(0, 255, 0), 2)  return img,objectInfo   if \_\_name\_\_ == "\_\_main\_\_":  cap = cv2.VideoCapture(0)  cap.set(3, 640)  cap.set(4, 480)  while True:  success, img = cap.read()  result, objectInfo = getObjects(img,0.5,0.2)  #print(objectInfo)  cv2.imshow("output", img)  cv2.waitKey(1) |
| --- |

**Figure 5: Object Detection class used to detect people and create a bounds box**

| from ObjectDetectionModule import \*  cap = cv2.VideoCapture(0) cap.set(3, 640) cap.set(4, 480)  while True:  success, img = cap.read()  result, objectInfo = getObjects(img,0.5,0.2,objects=['person'])  #print(objectInfo)  cv2.imshow("output", img)  cv2.waitKey(1) |
| --- |

**Figure 6: Main Module class that utilizes Object Detection class**