**CS437: Internet of Things**

**Lab1 Part1**

**Name: Kinga Bullert-Niesen, Zachary Moulton, Robert Coffey**

**NetID: kingaab2, moulton6, rc37**

**Late days used: 0**

**Video Link: Can be found in the folder** [**here**](https://drive.google.com/file/d/1DQgUllMsk4ttfjI1ZIm8OGx0O_UlqCc4/view?usp=sharing) **and also on Youtube** [**here**](https://www.youtube.com/watch?v=cbFXjeE15Bs)

A picture containing text, scoreboard, cabinet

Description automatically generated**Topology:**

PWM P0

MOTOR 1

PWM P13 D4

Servo

PWM P12 D5

MOTOR 2

MOTOR 3

PWM P8 D11

MOTOR 4

PWM P9 D15

ADC A5 A6 A7

Grayscale Sensor

A close-up of a circuit board

Description automatically generated with medium confidenceD8 D9

Ultrasonic Sensor

Camera

Table

Description automatically generated

**Design Consideration**

The following considerations were made when preparing the solution to this Lab.

**Chassis Assembly:**

The initial functional tests were successfully performed with the motors, servo, and sensors. The assembly instructions were followed without issue and care was taken to gather the wiring through the center of the car to keep it groomed. Pictures of the assembled car can be seen as follows.

**A picture containing floor, indoor

Description automatically generated**



**A toy car on a wood floor

Description automatically generated with low confidenceA toy car on a wood floor

Description automatically generated with low confidence**

The servo required some physical adjustment and tuning to secure that the direction of the ultrasonic sensor was correct and the 180 degree scan was performed correctly, otherwise no further optimizations were needed.

The camera ribbon was securely taped down to avoid interference with the ultrasonic sensor. The module was mounted on the front of the car. Tie wraps were used rather than tape to avoid potential interference with the board components.

**SW implementation**

The SW implementation consists of a number of Python functions shown in the following diagram.

Letter

Description automatically generated

**Naive Mapping:**

In order to map the environment in front of the car the ultrasonic sensor was used.

Ultrasonic sensor

The ultrasonic sensor performs continuous 180 degree scans of the area in front of the car and returns a list of 10 numbers for each scan. Each number in the list represents points with an 18 degree separation distributed across the 180 degree range, where the first point is the left-most point when facing in the direction of the car and 10th point the right-most point.

The values of each element in the list reflect the presence and distance of an obstacle in that specific direction. The numbers can be 2, 1 or 0 where 2 represents no obstacles detected, 1 represents object detected at a distance of 10-35 cm from the sensor and 0 represents object detected close to the sensor ie. <10 cm.

Mapping

Each scan result is analyzed in the “check\_scan” function to produce a rudimentary map, represented by the “blocked\_state” dictionary, which indicates if obstacles are found in the left, center(front) or right areas around the car.

This map is then used as input to further decision making regarding the movement and direction of the car.

**Naive Self-driving:**

Based on the inputs from the previous section the car is able to drive and take the needed actions to avoid obstacles.

Setting a destination

To start with, the car needs somewhere to go. For this purpose the destination is defined simply as infinity in the direction the car is facing when it starts. The car continually attempts to return to this direction.

Planning

Once each scan has been performed and any potential obstacles are detected. The car plans its next action.

The decision regarding the next step is made in the function “decide\_on\_action”, where three inputs are used to make the needed decision, these inputs are as follows:

* “blocked\_state” which informs if there are obstacles to the left, center, or right of the car’s current driving direction.
* “direction” which contains the current driving direction of the car in relation to the wanted destination (ie. the car’s initial driving direction)
* “distance \_counter” which counts how many times the car has moved forward since the last turn

Based on these inputs the car action is decided. There are three possible actions: continue moving forward, turn right and turn left.

Decision logic

The decision logic can be shortly summarized as follows.

* If the car is moving towards the direction of the destination, then keep going.

* If the car finds an obstacle, then turn to avoid the obstacle and travel a minimum distance forward before attempting to turn back towards the destination.

* Continue attempting to turn back towards the destination, using the scan output to check the way is clear.

* Continue avoiding obstacles and turn as needed to avoid them

Executing a turn

Turns are performed by turning the car 90 degrees in the wanted direction. The 90 degree angle is realized within the “turn” function, by executing a turn for a specific period of time, (around 1 second). The turning timer was made specific for right and left turns due to slight differences in the car behavior when turning in the different directions and the need for the 90 degree turn to be precise.

Moving forward

The car is moved forward by the “move\_forward” function which calls the respective pi-car function with a speed parameter.

|  |  |
| --- | --- |
| **Name** | **Contribution** |
| Kinga | Code review & documentation |
| Zachary Mouton | Code review |
| Robert Coffey | Code development, video and documentation |