Hochschule Ravensburg-Weingarten

University of Applied Sciences

Task 3: Remote Control

Sr No	Student Names	Matriculation Numbers
1	Arjun Ramwshbhai Beladiya	28744163

Guided by:
Prof. Markus Pfeil

Task A)Your Task is to create a python program that will move the robot forward until it comes close to an obstacle and stop about 10cm from it. Instead of calling the python script from the shell it should be triggered from the index.html page using a checkbox. Review the preparation exercise and modify the webserver.js script to execute a python script instead of toggling the LED. When the robot has stopped the website should indicate the distance from the obstacle as well as the distance travelled by each wheel and un-check the control box used to start the movement. Using another control box (copy and rename the corresponding parts in the webserver and index files) you should now execute a second script that will move the robot backward 20cm and turn a random angle, then call the first script again to move until encountering an object. Use a third control box it initiate a random movement by continuously initiating scripts one and two until the control box is unchecked (the current script will be finished then and no new one initiated) During the movement output the distance travelled by each wheel and the distance from the object stopped at on the webpage (one line per movement)

Task B) Summing up, firstly modify the index.html page with the html tags for displaying the distance and the checkboxes for initiating the movement to reflect the new desired behaviour.

Index.html file:

```
<!DOCTYPE html>
<html>
<body>

<h1>Control Robot</h1>

<!-- Original LED control -->
<p><label><input type="checkbox" id="light"> LED Light</label></p>

<!-- New controls for robot -->
<p><label><input type="checkbox" id="forward"> Move Forward Until
   Obstacle</label></p>
<p><label><input type="checkbox" id="backward"> Move Backward 20cm +
   Turn</label></p>
<p><label><input type="checkbox" id="random"> Random Movement</label></
   p>

<h2>Status:</h2>
<div id="statusLog"></div>

<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io(); // Connect to server

  window.addEventListener("load", function () {
    // LED Control
    var lightbox = document.getElementById("light");
    lightbox.addEventListener("change", function () {
      socket.emit("light", Number(this.checked));
    });

    // Forward Movement
    var forwardBox = document.getElementById("forward");
    forwardBox.addEventListener("change", function () {
      socket.emit("moveForward", this.checked);
    });

    // Backward + Turn
    var backwardBox = document.getElementById("backward");
    backwardBox.addEventListener("change", function () {
      socket.emit("moveBackward", this.checked);
    });

    // Random Movement
    var randomBox = document.getElementById("random");
    randomBox.addEventListener("change", function () {
      socket.emit("randomMove", this.checked);
    });
```

```
  });

  // Sync LED from server
  socket.on("light", function (data) {
    document.getElementById("light").checked = data;
  });

  // Log status messages
  socket.on("status", function (message) {
    const log = document.getElementById("statusLog");
    const p = document.createElement("p");
    p.textContent = message;
    log.appendChild(p);
  });
</script>

</body>
</html>
```

Task c) Secondly modify the webserver.js file to change the behaviour of the checkboxes to run the python script that will trigger the movement.

webserver.js file

```
var http = require('http').createServer(handler);
var fs = require('fs');
var io = require('socket.io')(http);
var Gpio = require('onoff').Gpio;
var { spawn } = require('child_process');

var LED = new Gpio(7, 'out');
var pushButton = new Gpio(5, 'in', 'both');

http.listen(8080);

function handler (req, res) {
  fs.readFile(__dirname + '/public/index.html', function(err, data) {
    if (err) {
      res.writeHead(404, {'Content-Type': 'text/html'});
      return res.end("404 Not Found");
    }
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}

io.sockets.on('connection', function (socket) {
  var lightvalue = 0;

  pushButton.watch(function (err, value) {
    if (err) {
      console.error('There was an error', err);
      return;
    }
    lightvalue = value;
    socket.emit('light', lightvalue);
  });

  socket.on('light', function(data) {
    lightvalue = data;
    if (lightvalue != LED.readSync()) {
      LED.writeSync(lightvalue);
    }
  });
  socket.on('moveForward', function (active) {
    if (active) {
      const process = spawn('python3', ['T_3_Forward.py']);

      process.stdout.on('data', function (data) {
        socket.emit('status', data.toString()); // send to browser
      });

      process.stderr.on('data', function (data) {
```

```
        socket.emit('status', "Error: " + data.toString());
      });

      process.on('close', function () {
        socket.emit('moveForward', false); // uncheck forward box
        socket.emit('status', '    Forward motion complete');
      });
    }
  });
  socket.on('moveBackward', function (active) {
    if (active) {
      const process = spawn('python3', ['T_3_Backward.py']);

      process.stdout.on('data', function (data) {
        socket.emit('status', data.toString());
      });

      process.stderr.on('data', function (data) {
        socket.emit('status', "Error: " + data.toString());
      });

      process.on('close', function () {
        socket.emit('moveBackward', false); // uncheck backward box
        socket.emit('status', '    Backward + turn complete');
      });
    }
  });

  let randomProcess = null;

  socket.on('randomMove', function (active) {
    if (active) {
      randomProcess = spawn('python3', ['T_3_random_motion.py']);

      randomProcess.stdout.on('data', function (data) {
        socket.emit('status', data.toString());
      });

      randomProcess.stderr.on('data', function (data) {
        socket.emit('status', "Error: " + data.toString());
      });

      randomProcess.on('close', function () {
        socket.emit('randomMove', false);
        socket.emit('status', '    Random loop ended');
      });
    } else {
      if (randomProcess) {
        randomProcess.kill('SIGINT'); // graceful stop
        randomProcess = null;
       }
    }
  });
});

process.on('SIGINT', function () {
  LED.writeSync(0);
  LED.unexport();
```

```
  pushButton.unexport();
  process.exit();
});
```

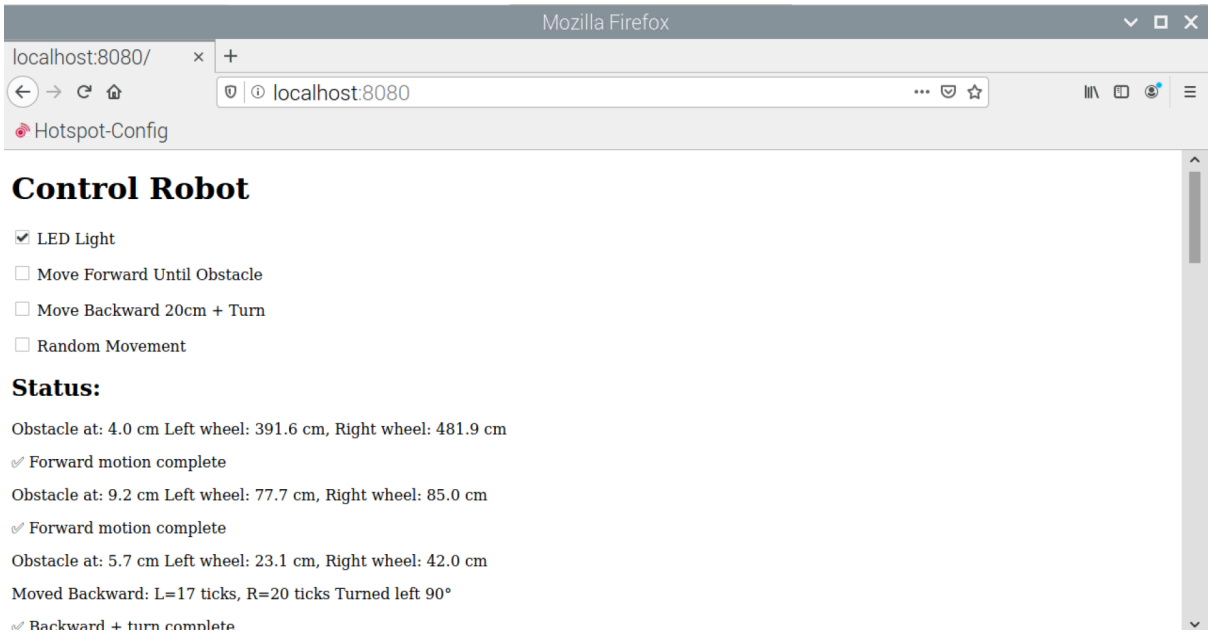

Figure 1: Result

D) Thirdly create the necessary phyton scripts.

1) Forward Motion untill Obstacle

```
import RPi.GPIO as GPIO
import time

# GPIO setup
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

# Motor and Encoder Pins
Motor1_PWM = 18; Motor1_IN1 = 17; Motor1_IN2 = 22; Encoder1 = 16
Motor2_PWM = 19; Motor2_IN1 = 24; Motor2_IN2 = 4;  Encoder2 = 23
TRIG = 25; ECHO = 27

# Setup
GPIO.setup([Motor1_PWM, Motor1_IN1, Motor1_IN2, Motor2_PWM, Motor2_IN1,
    Motor2_IN2], GPIO.OUT)
GPIO.setup([Encoder1, Encoder2, ECHO], GPIO.IN)
GPIO.setup(TRIG, GPIO.OUT)

pwm1 = GPIO.PWM(Motor1_PWM, 100)
pwm2 = GPIO.PWM(Motor2_PWM, 100)
pwm1.start(0)
pwm2.start(0)

# Encoder tick counters
left_ticks = 0
right_ticks = 0

def encoder1_cb(channel):
    global left_ticks
    left_ticks += 1

def encoder2_cb(channel):
    global right_ticks
    right_ticks += 1

GPIO.add_event_detect(Encoder1, GPIO.BOTH, callback=encoder1_cb)
GPIO.add_event_detect(Encoder2, GPIO.BOTH, callback=encoder2_cb)

# Distance measurement
def read_distance():
    GPIO.output(TRIG, False)
    time.sleep(0.01)
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)
    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()
    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()
    return (pulse_end - pulse_start) * 17150  # cm

def move_forward(pwm_base=50):
    pwm1.ChangeDutyCycle(pwm_base)
```

```
    GPIO.output(Motor1_IN1, GPIO.HIGH)
    GPIO.output(Motor1_IN2, GPIO.LOW)
    pwm2.ChangeDutyCycle(pwm_base + 10)
    GPIO.output(Motor2_IN1, GPIO.HIGH)
    GPIO.output(Motor2_IN2, GPIO.LOW)

def stop():
    pwm1.ChangeDutyCycle(0)
    pwm2.ChangeDutyCycle(0)
    GPIO.output(Motor1_IN1, GPIO.LOW)
    GPIO.output(Motor1_IN2, GPIO.LOW)
    GPIO.output(Motor2_IN1, GPIO.LOW)
    GPIO.output(Motor2_IN2, GPIO.LOW)

try:
    move_forward()
    while True:
        dist = read_distance()
        if dist < 10:
            stop()
            break
        time.sleep(0.05)

    wheel_circum = 21  # cm (adjust to your robot)
    left_dist = left_ticks * wheel_circum / 20  # example: 20 ticks/rev
    right_dist = right_ticks * wheel_circum / 20

    print(f"Obstacle at: {dist:.1f} cm")
    print(f"Left wheel: {left_dist:.1f} cm, Right wheel: {right_dist:.1
       f} cm")

finally:
    pwm1.stop()
    pwm2.stop()
    GPIO.cleanup()
```

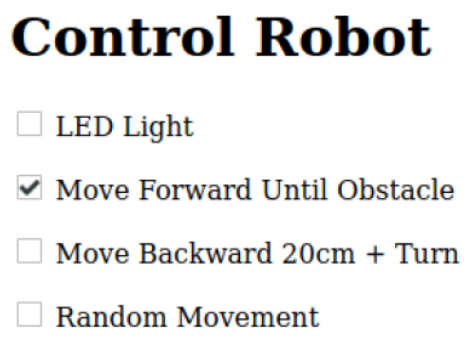

Figure 2: Result

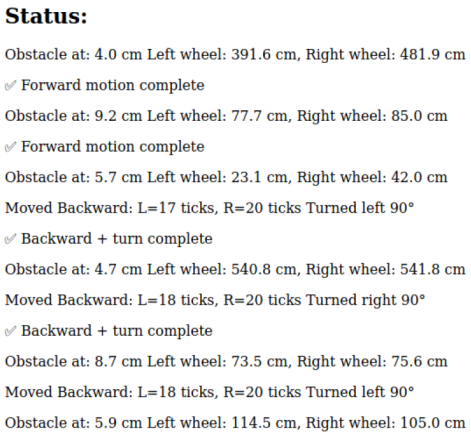

Figure 3: Result

2) Backward + Turn

```
import RPi.GPIO as GPIO
import time
import random
import subprocess

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

# Pins
Motor1_PWM = 18; Motor1_IN1 = 17; Motor1_IN2 = 22; Encoder1 = 16
Motor2_PWM = 19; Motor2_IN1 = 24; Motor2_IN2 = 4;  Encoder2 = 23

GPIO.setup([Motor1_PWM, Motor1_IN1, Motor1_IN2, Motor2_PWM, Motor2_IN1,
    Motor2_IN2], GPIO.OUT)
GPIO.setup([Encoder1, Encoder2], GPIO.IN)

pwm1 = GPIO.PWM(Motor1_PWM, 100)
pwm2 = GPIO.PWM(Motor2_PWM, 100)
pwm1.start(0)
pwm2.start(0)

back_left_ticks = 0
back_right_ticks = 0

def encoder1_cb(channel):
    global back_left_ticks
    back_left_ticks += 1

def encoder2_cb(channel):
    global back_right_ticks
    back_right_ticks += 1

GPIO.add_event_detect(Encoder1, GPIO.BOTH, callback=encoder1_cb)
GPIO.add_event_detect(Encoder2, GPIO.BOTH, callback=encoder2_cb)

def backward(pwm_base=50, target_ticks=20):
    pwm1.ChangeDutyCycle(pwm_base)
    GPIO.output(Motor1_IN1, GPIO.LOW)
    GPIO.output(Motor1_IN2, GPIO.HIGH)
    pwm2.ChangeDutyCycle(pwm_base + 10)
    GPIO.output(Motor2_IN1, GPIO.LOW)
    GPIO.output(Motor2_IN2, GPIO.HIGH)
    while back_left_ticks < target_ticks and back_right_ticks <
       target_ticks:
        time.sleep(0.01)

def stop():
    pwm1.ChangeDutyCycle(0)
    pwm2.ChangeDutyCycle(0)
    GPIO.output(Motor1_IN1, GPIO.LOW)
    GPIO.output(Motor1_IN2, GPIO.LOW)
    GPIO.output(Motor2_IN1, GPIO.LOW)
    GPIO.output(Motor2_IN2, GPIO.LOW)

def turn_random():
    direction = random.choice(["left", "right"])
```

```
    ticks_needed = 15  # adjust for 90 degrees

    if direction == "left":
        GPIO.output(Motor1_IN1, GPIO.LOW)
        GPIO.output(Motor1_IN2, GPIO.LOW)
        pwm2.ChangeDutyCycle(60)
        GPIO.output(Motor2_IN1, GPIO.LOW)
        GPIO.output(Motor2_IN2, GPIO.HIGH)
        count = 0
        last = GPIO.input(Encoder2)
        while count < ticks_needed:
            now = GPIO.input(Encoder2)
            if now != last:
                count += 1
                last = now
    else:
        GPIO.output(Motor2_IN1, GPIO.LOW)
        GPIO.output(Motor2_IN2, GPIO.LOW)
        pwm1.ChangeDutyCycle(60)
        GPIO.output(Motor1_IN1, GPIO.LOW)
        GPIO.output(Motor1_IN2, GPIO.HIGH)
        count = 0
        last = GPIO.input(Encoder1)
        while count < ticks_needed:
            now = GPIO.input(Encoder1)
            if now != last:
                count += 1
                last = now

    stop()
    print(f"Turned {direction} 90  ")

try:
    backward()
    stop()
    print(f"Moved Backward: L={back_left_ticks} ticks, R={
       back_right_ticks} ticks")
    time.sleep(0.5)
    turn_random()
    time.sleep(0.5)
    subprocess.call(["python3", "T_3_Forward.py"])

finally:
    pwm1.stop()
    pwm2.stop()
    GPIO.cleanup()
```

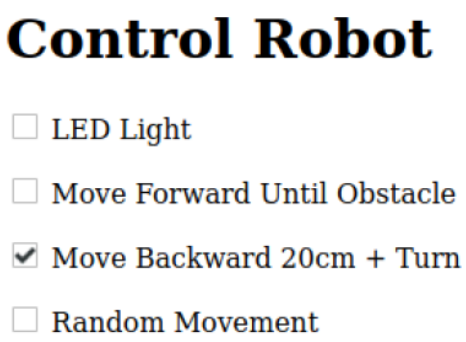

Figure 4: Result

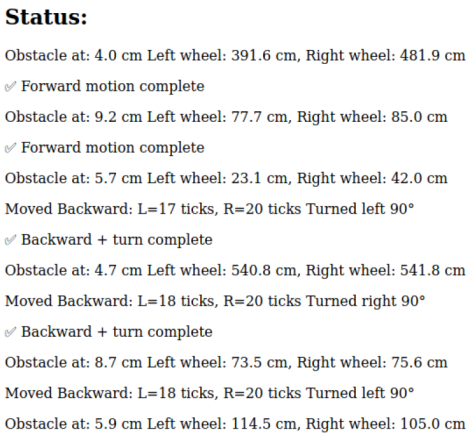

Figure 5: Result

3) Random Movement

```
import subprocess
import time

try:
    while True:
        subprocess.call(["python3", "T_3_Backward.py"])
        time.sleep(0.5)  # optional pause
except KeyboardInterrupt:
    print("Random loop stopped.")
```

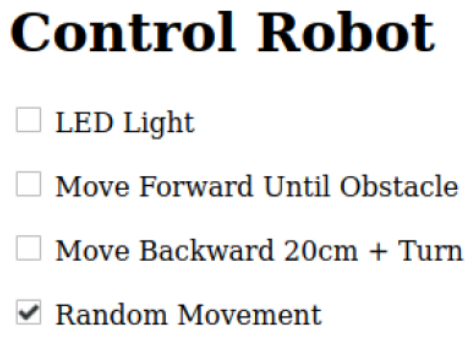

Figure 6: Result

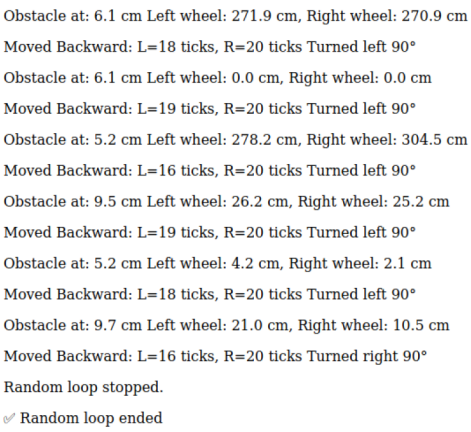

Figure 7: Result