POZNAŃ UNIVERSITY OF TECHNOLOGY

FACULTY OF CONTROL, ROBOTICS

AND ELECTRICAL ENGINEERING

TERM DESIGN PROJECT

PROJECT REPORT

PAWEŁ ROZENBLUT, 140361

pawel.rozenblut@student.put.poznan.pl

EWELINA ROBAKOWSKA, 140360

ewelina.robakowska@student.put.poznan.pl

JAKUB ZAMIATOWSKI, 140366

jakub.zamiatowski@student.put.poznan.pl

INSTRUCTOR:

Jan Wietrzykowski, M. Sc.

jan.wietrzykowski@put.poznan.pl

27-06-2021.

Contents

‘

1 Introduction

The aim of the project was to adapt the AlphaBot2-Ar platform for the purposes of maze traversing. The adaptation concerned both the hardware modifications (additional sensors and pin reconfiguration) and the development of purpose-specific software (robot control routines). Project completion involved construction of the maze as well, on which we could test the platform. The conceptual assumptions regarding the wheeled platform - modify it in such a way, that would render it capable of left-hand side wall following, detection of intersections and movement trajectory compensation.

2 Base platform - AlphaBot2-Ar

AlphaBot2-Ar robot kit includes a chassis (AlphaBot2-Base chassis) and an adapter board AlphaBot2-Ar. The platform itself is a two-wheeled robot with ball supports, on its bottom part. The upper part contains pass-through raster signal connectors (standard 2.54 [mm]; to access the Arduino pins), an LCD display, joystick and 2 PIN jumpers to connect/disconnect platform's peripherals' communication signal pathways. Aside from this, the platform is equipped with a number of sensors and chips for communication/control of different sensors, IO devices, motors etc. but most of those are unused in our case. The complete description of the platform can be found in [1]

Figure 1: AlphaBot2-Ar robot.

3 Platform modification - additional sensors

In order to adapt the Alphabot2-Ar for being capable of traversing the maze in the 'wall-follower mouse robot' configuration, the additional sensors were installed; the list of which, along with the placement description is presented in this section.

3.1 Waveshare 9523

The reflective module with IR sensor equipped with digital and analog output. It works with a voltage from 3 V to 5.3 V. It enables the detection of obstacles in the range from 0 to 30 mm.

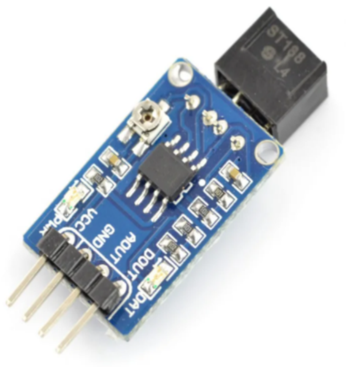

Figure 2: Waveshare 9523 sensor

3.2 Reflective module with IR sensor

The reflective module with IR sensor equipped with digital output. It works with a voltage from 3.3 V to 5 V. It enbales the detection of obstacles in the range from 2 to approx. 20 cm. It is based on LM393 system.

Figure 3: Reflective sensor

3.3 Additional sensor placement

The figure below presents the location of the listed above sensors on the platform.

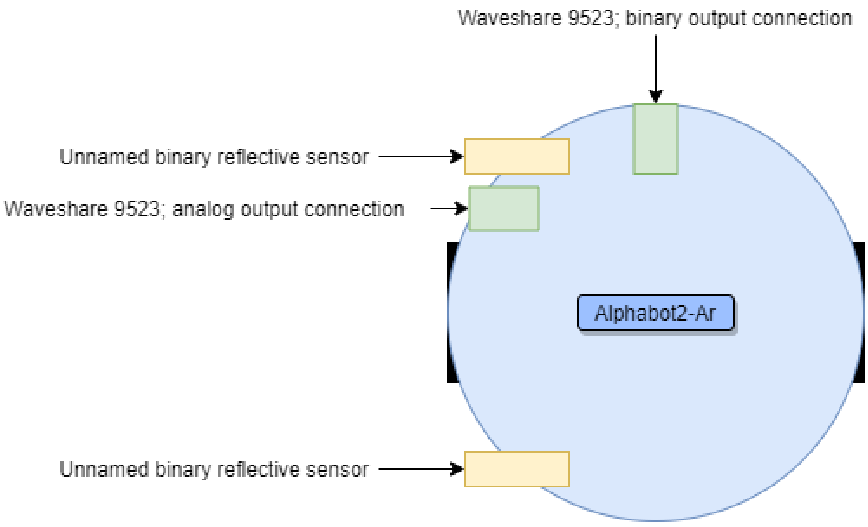

Figure 4: Alphabot2-Ar top part sensor placement

The Waveshare 9523 sensor on the upper part of the robot is responsible for front-wall detection. It is operating in the binary mode. The other Waveshare sensor, on the left side, is used for compensation in order to achieve straight movement movement of the robot (relative to it's desired path trajectory); (analog operation). Two other sensors are responsible for checking if the left side of the robot is 'free' - i.e. if there is no wall.

4 Maze

For the purpose of testing and checking the behaviour of robot we build a special maze. The construction is fully made with wood. The base is plywood whereas the walls are made with wooden rails. For the purpose of better accuracy of sensor the sides of the wall have been painted with white paint.

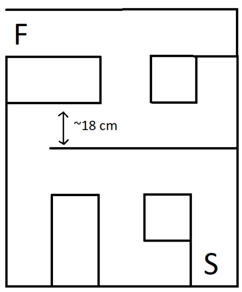

Figure 5: Maze illustrative schematic

Figure 6: Physical realisation of the maze

5 Software algorithm implementation

5.1 Core concept

The platform is supposed to follow the wall on its left side. Based on the signal received from the signals, it should detect whether:

1. There is wall on its left side
2. There is wall in front of it.

Additionally, it should compensate its movement trajectory on the basis of the readouts from the analog IR sensor. Based on those signals, it should follow the following instruction list:

1. If there is wall on the left, and no wall up front → go straight.
2. If there is no wall on the left → turn left.
3. If there is wall on the left, and there is wall up front → turn right.

5.2 Implementation

A number of Arduino-compliant C++ libraries were created for the purpose of motor and sensor handling; some of the code is based on the exemplary code pack provided by the manufacturer. The contents of said libraries is presented in the project's Github repository, to which the instructor has full access. In the listing below, the contents of the main .ino file are presented. As can be seen, the code is basically the implementation of the logic presented in the previous subsection, along with board setup configuration and additional helper functions.

Listing 1: Code Captions

```
#include "mobility.h"

#define COMPENSATION_LEFT A4 // sensor for path trajectory compensation analog value

#define DETECTION_LEFT 12 // binary IR sensor right
#define DETECTION_FRONT 3  // binary IR sensor front
#define COMPENSATION_CONDITIONAL_LEFT 4

#define WALL_TOO_FAR    600
#define WALL_TOO_CLOSE 400 //ADJUST!!

int left_speed = 20; // BASE 'POWER' DELIVERED TO THE LEFT MOTOR -> FOR COMPENSATION
int right_speed = 20; // BASE  'POWER' DELIVERED TO THE RIGHT MOTOR -> FOR COMPENSATION


int readout_left_compensate = 0;


bool iw_left = false;               // iw = is wall (present) -> readout memory placeholder
bool iw_front = false;              // same as above

int delay_value = 80;              // delay for 'driving forward' after detecting missing wall
const int delta_speed = 10;        // MAX DIFFERENCE IN POWER ON THE WHEELS
const int delta_compensation = 1; // ADD/SUBTRACT VALUE IN COMPENSATION ( < delta_speed)


// se
bool is_wall_left_back(){
    if(digitalRead(DETECTION_LEFT)==0) return true;
    else return false;
}

// se
bool is_wall_left_front(){
```

```
  if(digitalRead(COMPENSATION_CONDITIONAL_LEFT)==0) return true;
    else return false;
}
// se
bool is_wall_front(){
    if(digitalRead(DETECTION_FRONT)==0) return true;
    else return false;
}


// se
int read_compensator_left(){
  return analogRead(COMPENSATION_LEFT);
}

// compensate path trajectory, 2 pos relay with deadzone type control
void compensate() {
    readout_left_compensate = read_compensator_left();

    if(is_wall_left_front())
    {

      if(readout_left_compensate > WALL_TOO_FAR)
      {
        if(left_speed >(Speed - delta_speed)) left_speed -= delta_compensation;
        if(right_speed <(Speed +delta_speed)) right_speed += delta_compensation;

      }

      else if(readout_left_compensate < WALL_TOO_CLOSE)
      {

        if(left_speed <(Speed+delta_speed)) left_speed += delta_compensation;
        if(right_speed >(Speed -delta_speed)) right_speed -= delta_compensation;
      }

      else
      {
        left_speed = Speed;
        right_speed = Speed;
      }
    }

}



// MCU peripheral and interface setup
void setup()
{
  Speed = 40;

  delay(3000);
  Serial.begin(115200);


  pinMode(PWMA, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(AIN1, OUTPUT);
  pinMode(PWMB, OUTPUT);
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);


  pinMode(DETECTION_LEFT, INPUT);
  pinMode(DETECTION_FRONT, INPUT);
  pinMode(COMPENSATION_LEFT, INPUT);
  pinMode(COMPENSATION_CONDITIONAL_LEFT,INPUT);

  SetSpeeds(0, 0);
}

// Main program loop - wall follower algorithm
void loop()
{
  iw_left = is_wall_left_back();
  iw_front = is_wall_front();
  if(!iw_left){
    delay(delay_value); //keep driving forward for a while
    m_stop();
    m_ninety_left();    // turn left
```

```
        SetSpeeds(Speed,Speed);

        while(!is_wall_left_back()){
        if(is_wall_left_front()){
          compensate();
          SetSpeeds(left_speed,right_speed);
          }
      }
    }

  if(iw_left){

    if(iw_front) {
      m_stop();
      m_ninety_right();
      m_stop();
    }
    else {
      compensate();
      SetSpeeds(left_speed,right_speed);
    }
  }

}
```

5.3 Test results

The obtained results are not perfect, but satisfactory - the robot is able to traverse the maze (albeit with 'outside' help in cases when it gets stuck on the wall).

5.4 Conclusions

The objective has been partially realised. Some of the fixes that could have been applied to the platform occured to the authors after the project completion, for example - to utilise compensation routine shut-off based on both left-hand side binary IR sensors, instead of just one, or to utilise 'stuck' detection functionalities based on platform's built-in sensors.

6 Summary

Overall the project has been realised somewhat successfully. The presented hardware configuration has been implemented on the very last days before handing-in the project; before that, the authors tried a number of different approaches, with various results

7 References

1. Alphabot2-Ar documentation page. `https://www.waveshare.com/wiki/AlphaBot2-Ar`
2. Project repository - acces upon invitation `https://github.com/miodine/alphabot2Ar-maze_runner_software`