BIP Embedded Systems

Report

Robot@FactoryLite HW

06.01.2023 - 13.01.2023

Group 9:

Sophie Mießner 5046830

Josephina Ochynski 5096286

Lizhong Liu 418082

Bragança , January 31, 2023

Abstract

This paper is a documentation of our work during the the BIP course, Embedded Systems. Where we worked on the Robot@FactoryLite (Robot@FactoryLite) challenge. In the following pages we will state details on how we assembled our Robot, Implemented our Code and the challenges we faced while completing our task.

List of Abbreviations

BIP Blended Intensive Programme

Robot@FactoryLite Robot@FactoryLite

HSB Hochschule Bremen / City University of Applied Sciences

ipb Instituto Politécnico de Bragança

Hanze Hanzehogeschool Groningen / University of Applied Sciences

1 Introduction

This report describes our approach on the Robot@FactoryLite challenge which is part of the Embedded Systems course, provided by the Blended Intensive Programme (BIP) at the Polytechnic Institute of Bragança.

There are three universitys which are participating in the BIP: Hanzehogeschool Groningen / University of Applied Sciences (Hanze), Hochschule Bremen / City University of Applied Sciences (HSB) and Instituto Politécnico de Bragança (ipb).

The task included splitting into groups of max. four members. In our group we have the following three members:

1. Sophie from HSB
2. Josephina from HSB
3. Liu from Hanze

Since november we were prepared with weekly lessons and learned how to work with robots so we can participate in the challenge between the 6th and 13th of January 2023 at the ipb in Bragança .

The motivation behind learning about a topic using challenges and competitions, is that it adds excitement to the learning process. Moreover, this course was designed for students form different universities and technical backgrounds. A Fact that meant we had to navigate communicating on an international level, ultimately improving our soft skills.

We prepared our hardware-in-the-loop, using the SimTwo simulation environment, to simulate the robot, before travelling to Bragança. Plus we also prepared the finite state machine, which can be found here or `https://github.com/sophiemie/RAF_Simulator#readme`. Upon arriving in Bragança we transfered everything to the real robot prototype. A Prototype we build using the tools and materials provided by the ipb.

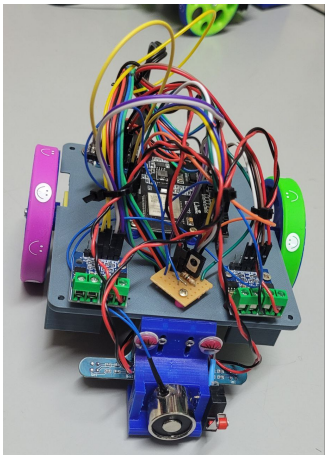


Figure 1: Robot Prototype

2 Related Work

During our virtual component of the Embedded Systems course, our Professors provided us with material. That helped us getting a better understanding of the parts our Robot has [Mar22a], what they do and how they communicate together. Moreover, explanations about the characteristics of a LineFollower Algorithm, could be found in this book, which was recommended to us in our lectures [Mor18]. A fact that gave us a better understanding about our example code.

We decided that the best approach to determine the behaviour of our robot was to use a state machine. A Topic that we also covered in our lectures [Mar22b].

Another task that we struggled with was the Simulation. Since the SimTwo Simulation software was new to us, we found a paper that explained the way it was working, why it was crated and which features it has.[Cos+11]

3 Implementation

3.1 The tasks

The following tasks were to be completed by the robot to as part of the challenge: Table 1

Description	Points
Follows the line correctly	5
Picks one box and succesfully places it in the correct destination	5
Picks another box and succesfully places it in the correct destination	10
Picks another box and succesfully places it in the correct destination	10
Picks another box and succesfully places it in the correct destination	10

Table 1: Tasks

To provide a better understanding of how these tasks were designed we want to elaborate a little on the idea behind the Robot@FactoryLite challenge.

The area used by our robot simulates a small factory floor. The dimensions are 1.7 x 1.2m. There are eight available machines (which are represented by the middle slots) and two warehouses (which are represented by the top slots and the slots below). The top slots are representing the source of the parts to be produced (the incoming warehouse) and the other slots are their final destination (the outgoing warehouse) [Bra+19]. Warehouses and machines are connected through tracks, forming a shape similar to a maze.

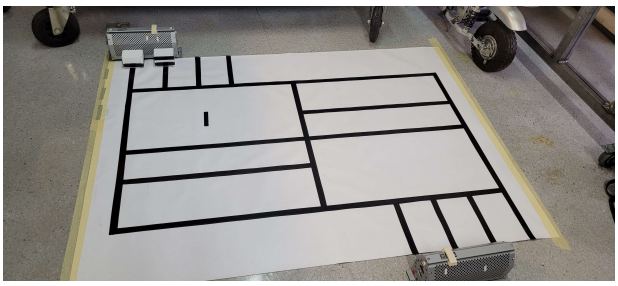


Figure 2: Factory Area

Focusing on our objective, which is to move boxes from one warehouse to another, using a magnet, the available machines (middle slots) are not going to play a significant role in our implementation. In other words the basic function of our Robot is to follow the black lines using a sensor, finding the correct way, pick up and place the boxes accurately.

3.2 Simulation

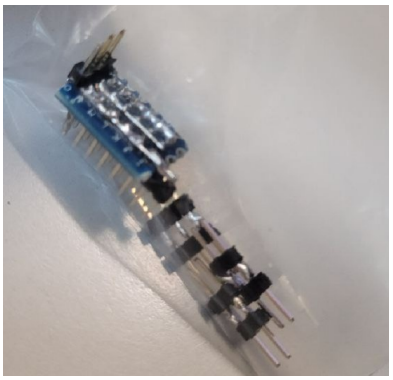
We are using SimTwo to simulate the Robot@FactoryLite. We can choose different scenes there, including one for this project. The scene encloses the robot and environment, needed for this project. Since everything that was written in the Simulator is in Pascal we use the hardware-in-the-loop approach for embedded systems. This implies that we programmed our esp32 microcontroller using Visual Studio Code and C/C++ and tested it using the simulated hardware.

After getting familiar with the SimTow environment we program the microcontroller. Everything we need for that is in the folder `raflite duinoHWLoopESP`. The first two tasks (state 1 to 8) from Table 1 are already implemented. So we just add the new states for the other three boxes in the file `control.cpp`. The code for the simulation, with the state machines we created and more description can be seen in our GitHub repository or `https://github.com/sophiemie/RAF_Simulator`.

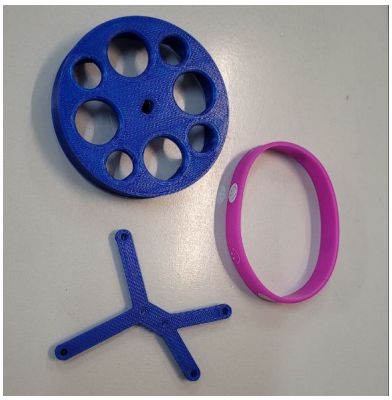
3.3 Hardware

Before we can run our robot we built it with the components on Figure 3, which we got from ipb. The wheels, casing and holders were printed with a 3D-printer (Figure 3c and Figure 3h).

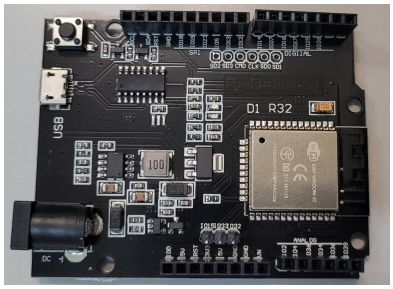
(a) Plastic Pins



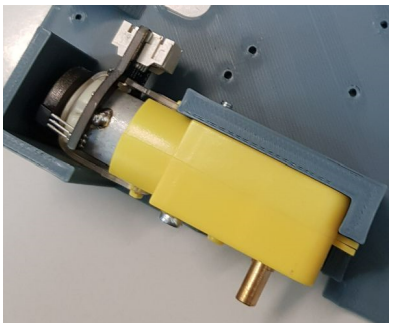
(b) Condenser



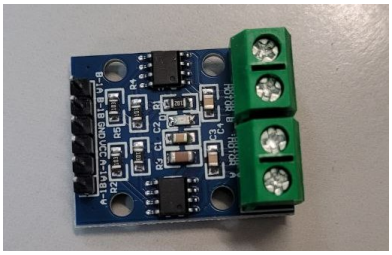
(c) Wheel Parts and Holder



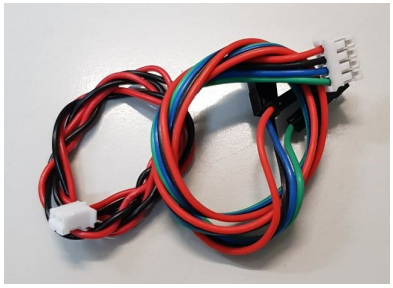
(d) Microcontroller



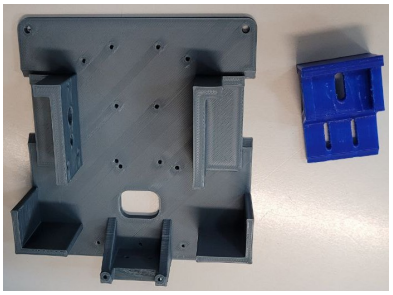
(e) Motor



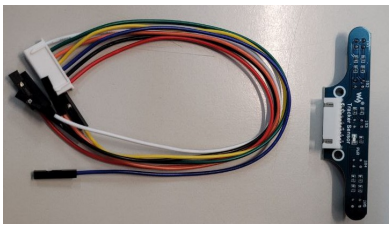
(f) Motor Driver x2



(g) Motor Cable

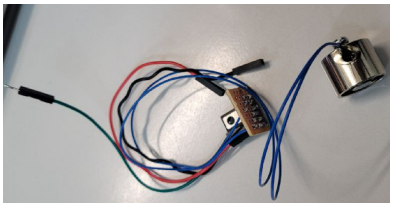


(h) Casing



(i) Line-Tracking-Sensor

(j) Switch



(k) Magnet

Figure 3: All components for the robot

3.3.1 Step 1

The first step included adding the motor for the wheels in the casing of our robot.

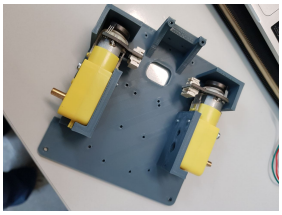


Figure 4: Step 1

3.3.2 Step 2

Secondly, we screwed a black casing that will hold the batteries at the bottom of the casing of out robot.

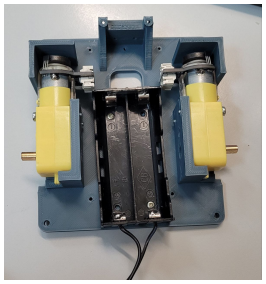
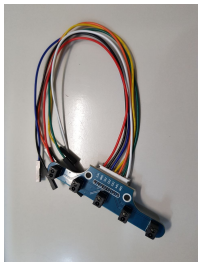


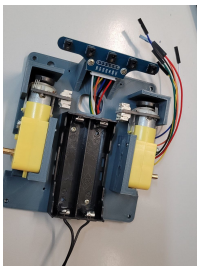
Figure 5: Step 2

3.3.3 Step 3

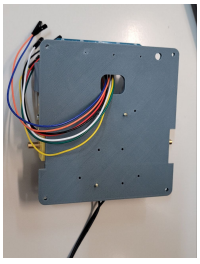
Next, we bolted our sensor. This will allow the robot to track the line, so that it can follow the correct path.



(a) Sensor with Cable



(b) Sensor on the robot

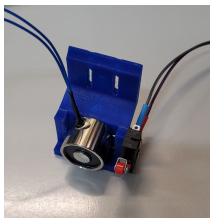


(c) Robot from behind

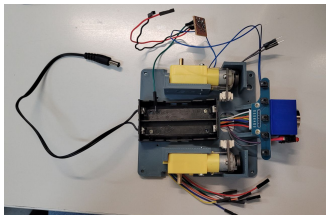
Figure 6: Step 3

3.3.4 Step 4

The point of this step was to provide our robot with a Magnet and a Switch. This allows our robot to pick up a box, carry it to the right destination and the ability to check whether the box is attached.



(a) Magnet and switch with holder

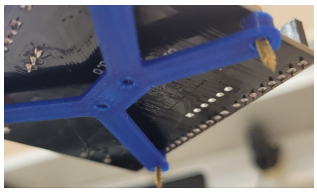


(b) New added component on robot

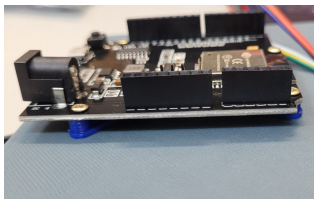
Figure 7: Step 4

3.3.5 Step 5

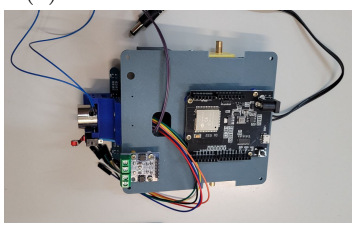
Step 5 included bolting the motor driver on top of the robot. That way it should be able to control the speed of our wheels and help the robot turn in the right direction.



(a) Plastic holder under the microcontroller



(b) Microcontroller on robot

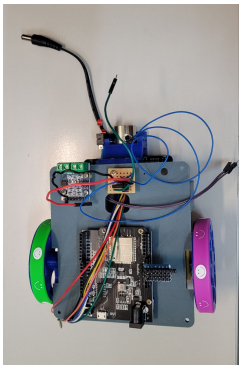


(c) Robot from behind

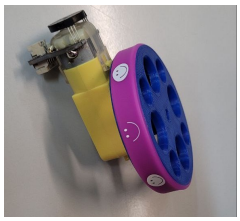
Figure 8: Step 5

3.3.6 Step 6

In the penultimate step we attached the wheels to the motors. In Figure 9a you can see the wheels on the robot. Figure 9b shows the wheel on the motor.



(a) Wheels on robot

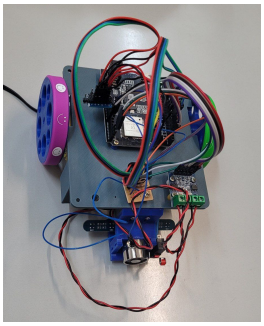


(b) Wheels on the motor

Figure 9: Step 6

3.3.7 Step 7

In this step we attached the wires so that every part of our robot is connected and can communicate. After that it will look like on Figure 10.



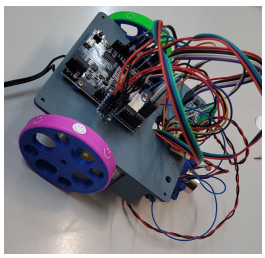
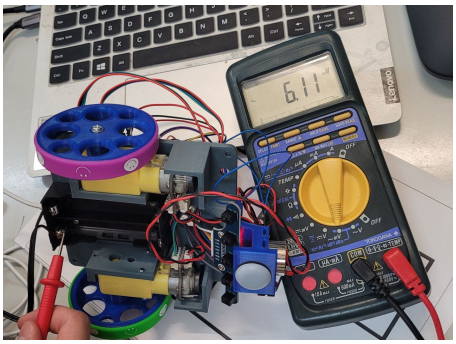


Figure 10: Step 7

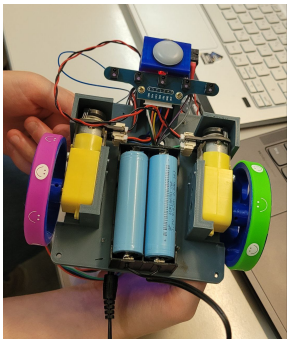
3.3.8 Step 8

The robot needs a power source before it can drive. So we had to find out if everything is connected properly before to putting the batteries in. We used the multimeter to test that (see Figure 11a). Unfortunately in this picture we are using the wrong settings, the right ones are on Figure 11b. After that we put the batteries in (see Figure 11c).



(a) Testing the batterie slots

(b) Settings multimeter

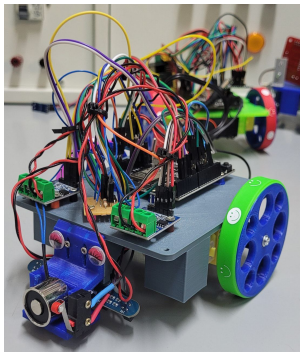


(c) Placed Batteries

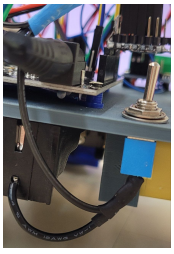
Figure 11: Step 8

3.3.9 Step 9

Accidentally we burned two motordrivers, which caused them to only work half way, meaning one motordriver was only able to control one wheel. This suggests using them both, one for each wheel Figure 12a. To spare the power of the batteries we added an on-of-switch (Figure 12b).



(a) Robot with two motordriver



(b) On-Off-Switch

Figure 12: Step 9

3.3.10 Run the robot

For our visit in Bragança we got a new code in the `raflitedu inoHWLoopESP` folder. Everything in there was already prepared for the real demonstration. We needed to change a few files and also needed to add new states in the `control.cpp` file. Like in subsection 3.2 the first eight states were already implemented. It didn't work for our robot so we needed to change the states a bit. We also have only the states for the first two boxes. The code can be found on `https://github.com/sophiemie/RAF`.

4 Numerical Examples

4.1 Changes in the original code

```
32 IRLine_t::IRLine_t()
33 {
34   IR_WaterLevel = 0;
35   IR_tresh = 500; // 300;
36   cross_tresh = 1; // 3;
37 }
```

Listing 1: Snippet from `IRLine.cpp`

First we had some problems with our robot about the counting of the crosses. It wouldn't count some crosses on the maze, which leads to fatal mistakes. So we changed the threshold values of the sensor in 1. We changed the variable `IR_tresh` from 300 to 500 and decreased the `cross_tresh` from 3 to 1. After that our robot counted the lines mostly right.

```
159 void setMotorLPWM(int PWM)
160 {
161   int PWM_max = 75;
```

```
177 void setMotorRPWM(int PWM)
178 {
179   int PWM_max = 75;
```

Listing 2: Snippets from `main.cpp`

In the main file in 2 we changed the PWM variables `PWM_max` of the motors to a lower number of 75, in line 161 and also in line 179, because we wanted to spare the motordrivers after we burned already two of them. That is also one of the reasons why our robot was slower than the other robots.

```
} else if(robot.state == 2 && robot.tis > 600) { // 100   20
      robot.rel_s = 0;                                    21
      robot.setState(3)                                   22
```

Listing 3: State 2 from `control.cpp`

In 3 we gave the robot more time to pick up the first box, so the magnet had enough time to connect with the box. After that, our robot picked up the box more often than before.

```
    } else if(robot.state == 7 && robot.tis > 1300) { // 2000  43
      robot.rel_s = 0; // we add that                         44
      IRLine.crosses = 0;                                     45
      robot.setState(8);                                      46
```

Listing 4: State 7 from `control.cpp`

In 4 we changed the time when the first box is placed at the new destination. The first value was to high and our robot would leave the maze.

```
// && robot.tis > 2000)                                  47
} else if(robot.state == 8 && robot.rel_s < -0.05)       48
{                                                        49
    IRLine.crosses = 0;                                  50
    robot.rel_theta = 0;                                 51
    robot.setState(9);                                   52
```

Listing 5: State 8 from `control.cpp`

We also changed the condition for the state change

from state 8 to 9 where the robot is supposed to drive backwards after bringing the first box to the destination. We are not asking the time anymore because of the change of our PWM values that wouldn't be enough time. Also for our new states we are using the variable who checks the distance instead of the time.

4.2 Our new states

We added the states 9 to 22 which are used to drive, pick up and place the second box. For the states where the robot is turning, like in state 9 and 16, we made the robot slower because otherwise the robot wasn't able to find the line. In state 16 we are turning the robot in the other direction so we set the value on `1.25`.

```
210     } else if (robot.state == 9) {  // Turn 180 degrees
211       robot.solenoid_state = 0;
212       //robot.setRobotVW(0,2.5);
213       robot.setRobotVW(0, -1.25);
```

Listing 6: State 9 from `control.cpp`

We needed one more state to get to the second box because we needed to change the Linefollowing of the robot, so the robot would turn to the right direction for this box. We are also using a lower speed for that way.

```
215 } else if (robot.state == 10) {  // turn left
216     robot.solenoid_state = 0;
217     robot.followLineLeft(IRLine, 0.1, -0.05);
218
219 } else if (robot.state == 11) {  // turn right
220     robot.solenoid_state = 1;
221     robot.followLineRight(IRLine, 0.1, -0.05);
```

Listing 7: State 10 and 11 from `control.cpp`

Our state 12 is also different from our prepared simulation. We decided not to use the followLine-functions when the robot is getting to the second box because our robot would always drive not straight to the box and the magnet would not pick up the box. So we turning the robot with the `setRobotVw()`-function. We also needed to change the condition for that. So now we need to ask about the angle of the turn instead of time.

```
} else if (robot.state == 12) {  // Go: Get second box
    robot.solenoid_state = 1;
    //robot.followLineLeft(IRLine, 0.06, -0.022);
    robot.setRobotVW(0.1, 1.25);
```

```
} else if(robot.state == 12 && robot.rel_theta > radians(60) ) {
    robot.rel_theta = 0;
    robot.rel_s = 0;
    robot.setState(13);
```

Listing 8: State 12 from `control.cpp`

For state 15 and also state 22 we are using different values than for state 8 for the part where the robot is driving backwards.

```
} else if(robot.state == 15 && robot.rel_s < -0.09)
{ // -0.04
    robot.rel_theta = 0;
    IRLine.crosses = 0;
    robot.setState(16);
```

Listing 9: State 15 from `control.cpp`

5 Future Work

One task that is the most noteworthy in this Section is the fact that even though our Robot can transport all four Boxes in the Simulation, during our Week in Bragança , we were not able to transfer everything correctly to our Robot Prototype. The reason for that was mainly time related. Assembling the robot was

very time consuming for us. Because we made the mistake of testing some Parts of it using a too high voltage. This resulted in replacing parts a few times and setting the maximum Speed of our robot low, to not damage the motordrivers any further. As a result we needed plenty of time to troubleshoot and figuring out the correct values each of our functions should receive to make the robot run smoothly on the factory area. Also, during our last run in the competition we noticed some small mistakes in our code but decided not to change anything since we were not able to test it anymore. This is a task that should be also done in the future. That was the part where our robot failed to put the second box at the right place.

Furthermore, as a future objective it could be possible to enhance the Robot and add an infrared sensor to it. That way it would be able to detect obstacles, after adjusting the code. That way the loop can be interrupted and use an algorithm to avoid the obstacle, while simultaneously finding the Line correct Line to follow again and continue with the original loop.

6 Acknowledgement

We are extremely grateful for the opportunity that was given to us by our Universities, due to their collaboration, to participate in the Blended Intensity Program. It was truly a rewarding experience, since we learned so much in such a short amount of time.
Furthermore this endeavor would not have been possible without the Polytechnic Institute of Bragança and Prof. José Lima who organized and supervised our work during the physical part of the Embbeded Systems course. Special thanks to Prof. Felipe Martins, who explained every question we had in detail, even if we asked the same thing twice.

Lastly, we would like to thank each other for pulling through and supporting each other. Even if we had some communication struggles in the beginning.

7 Contributions

Tasks	Author
Section: Abstract Section: Introduction Section: Implementation Section: Related Work Section: Acknowledgement Building the robot Fixing the robot Implement code in Bragança	Josephina
Section: Introduction Section: Implementation Section: Numerical Examples Building the robot Fixing the robot Implement code in Bragança Upload code on GitHub	Sophie
Fixing the Simulation in Bragança Building the robot Creating pictures of state mashines	Liu
Preparing the Simulation before we visit Bragança	All Authors

Table 2: Contributions of all Authors from Group 9

Bibliography

[Bra+19] Joao Braun et al. 2019. URL: https://core.ac.uk/reader/323508990.

[Cos+11] Paulo Costa et al. 2011. URL: https://www.academia.edu/80708613/SimTwo%5C_Realistic%5C_Simulator%5C_A%5C_Tool%5C_for%5C_the%5C_Development%5C_and%5C_Validation%5C_of%5C_Robot%5C_Software.

[Mar22a] Felipe N. Martins. 2022. URL: https://e1.pcloud.link/publink/show?code=kZQdg8ZOmlzpKkXUF4TWcCg1aiYMJfolder=4475096407&tpl=publicfoldergrid.

[Mar22b] Felipe N. Martins. 2022. URL: https://e1.pcloud.link/publink/show?code=kZQdg8ZOmlzpKkXUF4TWcCg1aiYMJfolder=4475096407&tpl=publicfoldergrid.

[Mor18] Francesco Mondada Mordechai Ben-Ari. 2018. URL: https://link.springer.com/book/10.1007/978-3-319-62533-1.