# Line follower

Borgioli Niccolo' and Rispo Veronica, ECS

29 aprile 2019

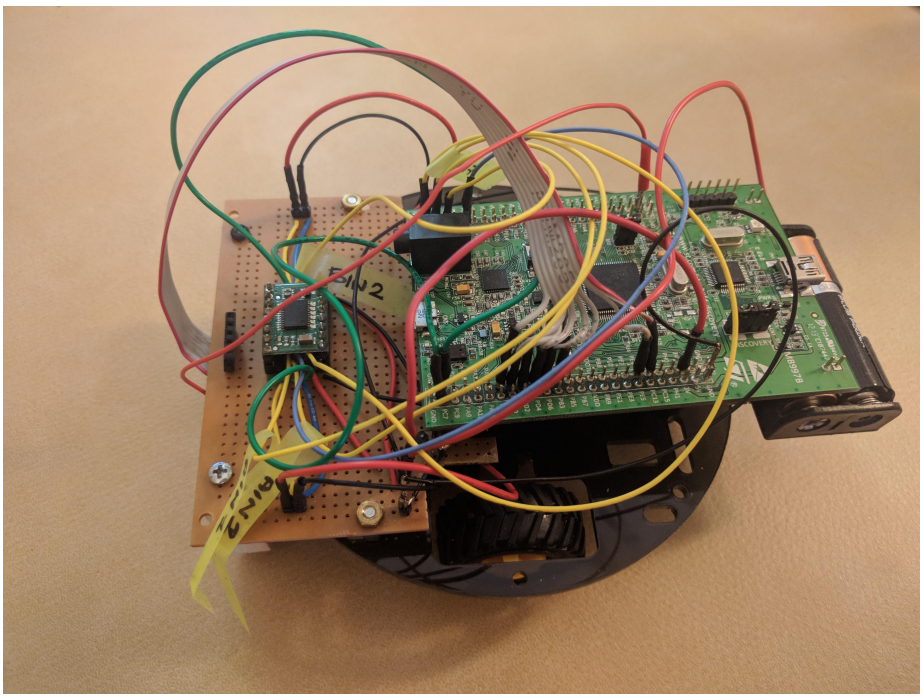

Figure 1: Line follower

# 1 INTRODUCTION

This report describes the design and development of the Design of Embedded Systems course project. In particular, it deals with the design of a machine that is able to follow a traced path (Figure 1).

# 2 REQUIREMENTS

The system, using data from the light sensors, must be able to control the movement of the wheels of the machine in order to:

- at system startup wheels are stopped
- reflectance sensors readings are performed continuously in order to detect the above surface color:
  - reading is requested to all sensors at same time by setting the pin to 1
  - a reading ends when the pin of the sensor goes to 0
  - when a sensor pin goes to 0 the relative time since reading has been commanded is registered
  - if a sensor does not returns a value within 60ms it goes in timeout
  - when all sensors have returned the reading (all pins have gone to 0 or in timeout) a new measure can start
  - if the response time of a sensor is lower than a LIGHT_THRESHOLD the color detected by that sensor is white
  - if the response time of a sensor is higher than a LIGHT_THRESHOLD the color detected by that sensor is black
  - if a sensor has gone in timeout the color detected by that sensor is assigned to be white
- if all sensors detect a white surface then stop both wheels
- if at least one sensor detects a black surface adjust movement of wheels in order to hold the black surface in the center. There are 8 sensors that are divided into two equal groups (left and right) depending on position respect to the center of the sensor array:
  - if the number of sensors that detected black surface is bigger on the left side of the array then stop the left wheel and move the right wheel
  - if the number of sensors that detected black surface is bigger on the right side of the array then stop the right wheel and move the left wheel
  - if equal number of sensors on left and right side detected a black surface then move both wheels at same speed
- if all sensors detect a black surface (horizontal black line) then stop both wheels

# 3 IMPLEMENTATION

The system has been implemented using an STM32F4DISCOVERY board [1] as control unit. To detect the line is used a reflectance sensor array QTR-8RC [2] of Pololu Robotics&electronics. The motors are controlled using a TB6612FNG Dual Motor Driver Carrier [3] connected to the discovery board and supplied by a battery pack in order to give power to motors.

## 3.1 SysML Diagrams

The SysML diagrams of the project are shown in Figure 2, Figure 3 and Figure 4.

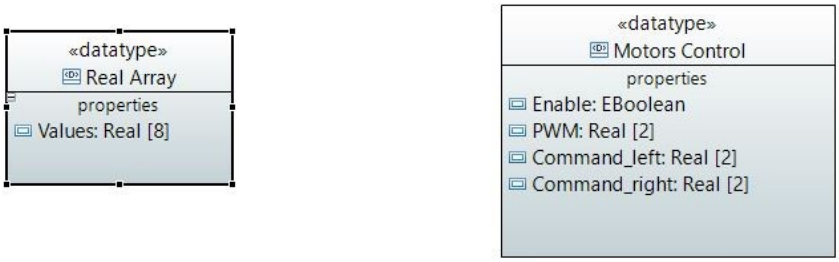


Figure 2: Type Definition Diagram

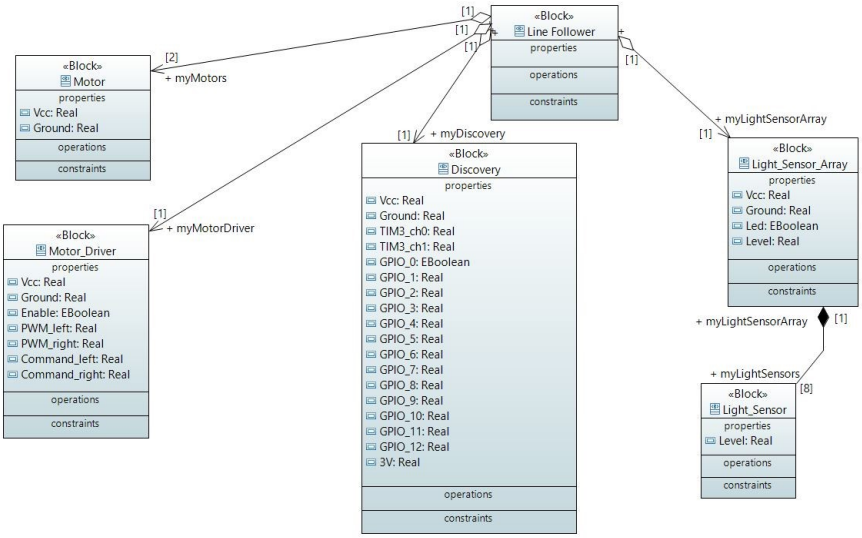


Figure 3: Block Definition Diagram

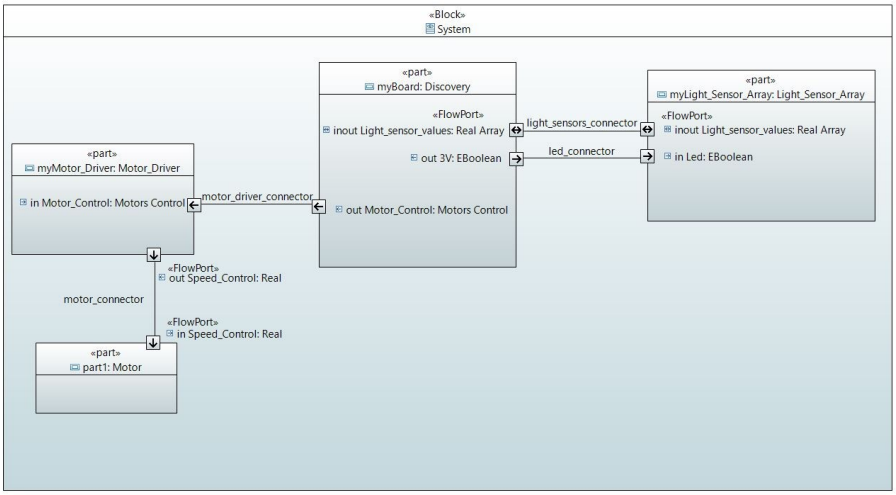


Figure 4: System Internal Block Diagram

## 3.2 Wiring

The reflectance sensor array QTR-8RC [2] is supplied by a battery pack and connected to the STM32F4DISCOVERY board [1]through nine pins. The first connection is between the 3V voltage pin supplied by the discovery and the sensor pin connected to the underlying LED, so that the same remains lit during the reading phase from the sensor array (in order to have a more precise reading). The other eight connections are between every sensor of the array and a GPIO pin of the discovery. The mode (input or output)of those GPIO pins is dinamically changed during the reading cycle.

The TB6612FNG Dual Motor Driver Carrier [3] is connected to the STM32F4DISCOVERY board [1]through seven pins. Two of those are with a channel of a timer initialized in order to provide the PWM signal to the motors. The other ones are instead connected to GPIO output pins: four in order to lead the motors (folle, forward, backward or locked)and the last one in order to enable the motor driver.

## 3.3 Code

The control of the motors and of the sensor is performed by two different dedicated tasks: MotorControlTask and CheckReadTask respectively.

In order to work these two tasks share a global variable (delta_sensor) which contains the last reading of the sensors (in ms). This variable is protected by a binary semaphore (delta_sensor_sem).

### 3.3.1 CheckReadTask

This task is in charge of starting the sensor, ask for a reading, wait for a response and then repeat these operations. To perform this the task is organized as a FSM where the actual state is stored in the sensor_mode variable.

```
1  TASK(CheckRead){
2    int i;
3    double delta;
4    int end;  //end flag
5
6    if(sensor_mode == SENSOR_START){
7      //INITIAL SENSOR SETUP
8      read_task_init(); //Setup interrupt handlers
9      sensor_mode = SENSOR_INIT;
10     system_time = 0;
11   }else if(sensor_mode == SENSOR_INIT){
12     InitLineSensor(); //Setup pins
13
14     //Put high all sensor pins
15     GPIO_SetBits(GPIOD, GPIO_Pin_1);
16     GPIO_SetBits(GPIOD, GPIO_Pin_3);
17     GPIO_SetBits(GPIOB, GPIO_Pin_4);
18     GPIO_SetBits(GPIOD, GPIO_Pin_5);
19     GPIO_SetBits(GPIOB, GPIO_Pin_6);
20     GPIO_SetBits(GPIOD, GPIO_Pin_7);
21     GPIO_SetBits(GPIOC, GPIO_Pin_10);
22     GPIO_SetBits(GPIOC, GPIO_Pin_12);
23
24
25     //Save system time
```

```
26      system_time = 0;
27      sensor_up_time = my_get_systime();
28
29      //Init pin readings and flags
30       for(i = 0; i < 8; i++){
31         led_ms[i] = 0;
32         led_flags[i] = 0;
33       }
34
35       sensor_mode = SENSOR_WAIT; //Put task in wait mode
36    }else if(sensor_mode == SENSOR_WAIT){
37
38      double actual_systick = my_get_systime();
39      if(actual_systick >= 60){
40        system_time = 0;
41        timeout_read = 1;
42      }
43      delta = actual_systick - sensor_up_time;  //Compute elapsed
        time from sensor pins up
44
45      if(delta >= DELTA_WAIT){
46
47        reference_time = my_get_systime();
48
49        //Set all pins as input
50        GPIO_InitStructure_LightSensors[0].GPIO_Mode=GPIO_Mode_IN;
51        GPIO_Init(GPIOB, &GPIO_InitStructure_LightSensors[0]);
52
53        GPIO_InitStructure_LightSensors[1].GPIO_Mode=GPIO_Mode_IN;
54        GPIO_Init(GPIOB, &GPIO_InitStructure_LightSensors[1]);
55
56        GPIO_InitStructure_LightSensors[2].GPIO_Mode=GPIO_Mode_IN;
57        GPIO_Init(GPIOD, &GPIO_InitStructure_LightSensors[2]);
58
59        GPIO_InitStructure_LightSensors[3].GPIO_Mode=GPIO_Mode_IN;
60        GPIO_Init(GPIOD, &GPIO_InitStructure_LightSensors[3]);
61
62        GPIO_InitStructure_LightSensors[4].GPIO_Mode=GPIO_Mode_IN;
63        GPIO_Init(GPIOD, &GPIO_InitStructure_LightSensors[4]);
64
65        GPIO_InitStructure_LightSensors[5].GPIO_Mode=GPIO_Mode_IN;
66        GPIO_Init(GPIOD, &GPIO_InitStructure_LightSensors[5]);
67
68        GPIO_InitStructure_LightSensors[6].GPIO_Mode=GPIO_Mode_IN;
69        GPIO_Init(GPIOC, &GPIO_InitStructure_LightSensors[6]);
70
71        GPIO_InitStructure_LightSensors[7].GPIO_Mode=GPIO_Mode_IN;
72        GPIO_Init(GPIOC, &GPIO_InitStructure_LightSensors[7]);
73
74        sensor_mode = SENSOR_READ;
75      }
76    }else if(sensor_mode == SENSOR_READ){
77      end = 1;
78      for(i = 0; i < 8; i++){
79        if(led_flags[i] == 0){
80          end = 0;
81          break;
82        }
83      }
84
85      if(system_time >= 60){
86        system_time = 0;
```

```
      timeout_read = 0;
      sensor_mode=SENSOR_INIT;
    }

    if(end == 1){
      //All sensors have returned value
      WaitSem(&delta_sensor_sem);
      for(i = 0; i < 8; i++){
        delta_sensor[i] = led_ms[i] - reference_time;
      }
      PostSem(&delta_sensor_sem);

      sensor_mode = SENSOR_INIT;
    }
  }
}
```

### 3.3.2 MotorControlTask

This task periodically reads from the shared variable delta_sensor the last value returned by the array of reflectance sensors and based on this determines the color of the material below each of these. Then depending on the colors detected commands different actions to the motors.

```
TASK(TaskMotorControl){
  double sensor_time[8];  //Local copy of decay time of light
    sensors
  int i;  //counter
  int left, right;  //counters of left and right sensor that
    detects black

  left = right = 0; //Initialize left and right black sensors
    counters


  //protect copy of delta_sensor to local sensor_time
  WaitSem(&delta_sensor_sem);
  for(int i=0; i < 8; i++)
    sensor_time[i] = delta_sensor[i];
  PostSem(&delta_sensor_sem);

  for(i = 0; i < 8; i++){
    if(sensor_time[i] > LIGHT_THRESHOLD){
      //Sensor detects black
      if(i < 4){
        //Left sensor
        left++;
      }else{
        right++;
      }
    }
  }

  if(left + right == 8){
    //All sensors detects black --> STOP (end of lap)
    breakleft();
    breakright();
  }else if(left > right){
    //Turn left
    forwardright();
    folleleft();
```

```
35    }else if(right > left){
36      //Turn right
37      follerright();
38      forwardleft();
39    }else if(left == 0 && right == 0){
40      //No info on the line (detected all white) --> STOP
41      breakleft();
42      breakright();
43    }else{
44      //Go straight forward
45      forwardleft();
46      forwardright();
47    }
48  }
```

# 4 CONCLUSIONS

Despite the problems detected during the testing phase, due to the malfunction of two elements of the light sensor array, the system has good performances. In fact, the two incriminated elements are both at the beginning of the array of light sensors (composed of eight elements) and then to solve the problem is sufficient:

- Follow a line less than the distance between three elements and wider than two
- Initially position the machine so that the line is about the center of the array
- Ignore the timeout of the two malfunctioning sensors

# AUTHORS

Niccolo' Borgioli took bachelor degree in Computer Engineering at University of Genoa, Italy, in 2016. Currently he is attending a Master Degree in Embedded Computing Systems at Scuola Superiore Sant'Anna (SSSA), Pisa, Italy. During Bachelor Degree he was co-founder and CEO of a start-up called LetIn.

Veronica Rispo took bachelor degree in Computer Engineering at University of Pisa, Italy, in 2015. Currently he is attending a Master Degree in Embedded Computing Systems at Scuola Superiore Sant'Anna (SSSA), Pisa, Italy.

# References

[1] https://www.st.com/en/evaluation-tools/stm32f4discovery.html

[2] https://www.pololu.com/product/961

[3] https://www.pololu.com/product/713