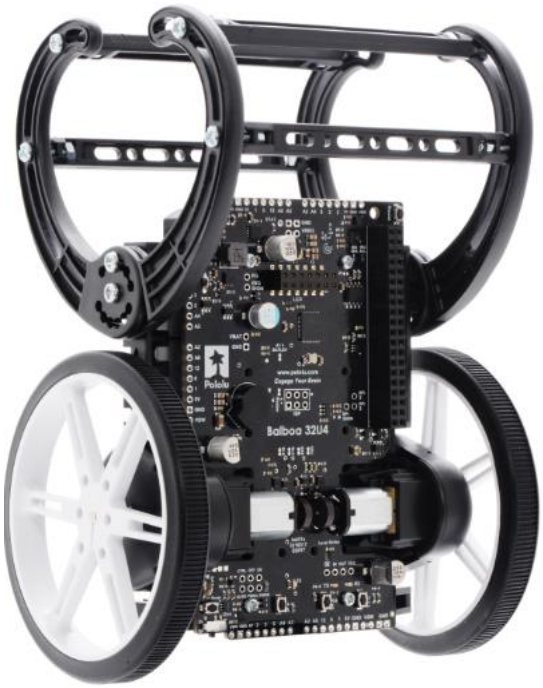
11/05/2023

Project Balboa 32U4

Return to base station



Wail ANKOURI 195181
Buch Keanu 16151
ECAM

Table of contents

1 Introduction

Concerning the various theme for the robot, my colleague and I decided to go for the “return to the base from a signal”. To do this, we had to use our knowledge in electronics and in the State-Space control as well as learn about infra-red signals. This is an interesting project because it can be used in various applications. For example, a vacuum cleaner, robots such as the Roomba cleaning robot and much more. In this report, we will explain how we met this challenge as well as its limitations.

2 Specifications

The specifications were defined with the with the clients (Mr De Bruyne and Mr Van Cauwenberghe). The following deliverables for the robot were expected:

1) The robot needed to be able to read an incoming IR signal, as well as having a base to generate it.
2) The robot needed to be able to return to the base (from where the IR signal was generated).
3) Additional temperature and smoke sensors were to be incorporated into the algorithm to act as a fire detection robot.

3 Material

In this section, we will show all the materials used for this project.

3.1 Ultra sonic sensor

To start off, to determine if the robot has reached the base or not, we added an obstacle at the base, which would be detected by a sensor. We used the ultrasonic sensor HCSR04 for this. This sensor is mounted on the front of the robot.

figure 1 : an ultrasonic sensor HCSR04.

The sensor characteristics are as follows:

- Maximum range of 4 m → sufficient to detect any object.
- Minimum distance of 2 cm → sufficient as the distance we used to detect an object was greater than 2 cm.
- Supply voltage of 5VDC → can be used on the 5V pins on the robot.
- Measuring angle of 15 degrees → small angles which ensured the detection of an object in front of the robot and not on the sides.

3.2 IR receiver

We used Infrared sensors to determine the direction in which the robot needed to travel. The KY-022 sensor was chosen as it possessed multiple qualities that we deemed necessary for a successful return to the base. The main advantages of these sensors are the following:

- Range of voltage (2.7V -> 5 V) → this allows us to use the 5V pin from the robot.
- The reception distance of the signal is 18 m.
- The reception angle of the signal is -+45 degrees.
- Ambient light filter of up to 500 lux (illuminance) → this allows us to filter the incoming infra-red signal in order to limit the ambient light which could interfere with it. For reference, a clear day will have about 400 lux. This allows us to use the robot during the day without too much interference.

As can be seen on figure 2, there are 3 pins. There is a ground (-), a power supply (+) and a pin that will process the received signal (S).

figure 2 : sensor KY-022

3.3 IR Transmitter

To determine the direction that the Robot needs to follow, four IR transmitters were used. these sensors were placed on a PCB that was installed on an Arduino, which acts as the base.

The reason we choose to send an infra-red signal is due to its wavelength of 920nm. As the wavelength of a signal increases it can travel longer distances, but with less information, therefore the longer the wavelength, the higher the probability of the signal going through solid matter. As a result, the receiver (KY-022) can theoretically read up to 18 meters. Naturally in practice that distance is considerably reduced.

The characteristics of the sensor are the following:

- Vf = 1.1 [V]: voltage drop
- If = 20 [mA]: forward current
- Wavelength = 920 nm

Furthermore, the IR signal is sent at a frequency of 38 Khz. The value of 38 khz was chosen because it is very rare to find such signals in nature which allowed us to reduce interferences coming from other sources of light, such as natural or artificial light. A signal modulated at 38 KHz is also easily intercepted by an IR receiver. The distance of the signal can vary based on the duty cycle, as was referenced in the datasheet produced for the Vishay IR receiver. The lower the duty cycle the longer the distance of reception[1]. The duty cycle in this case, was left at the default value of 30% but can be changed in the code using the macro IR_SEND_DUTY_CYCLE_PERCENT. The KY-005 can be seen in figure 3:

figure 3: Sensor KY-005

The sensor also possesses 3 pins, namely, a ground (-) (far right), ground (-) with a resistor (middle pin) and the pin that will send the signal (S) (left pin), which also acts as the power supply (+). The ground + resistor was not used in this case, as we did not need it. It is only needed when higher voltages are applied to the LED, therefore only 2 pins were actually connected.

To send the signal at 38khz, the IRremote Library was used, which uses interrupt timers on the Arduino to generate a PWM signal. The library allowed us to send and decode incoming signals with various protocols. The NEC protocol was used in this case as it is the most common one. The following illustration demonstrate how the signal is sent:

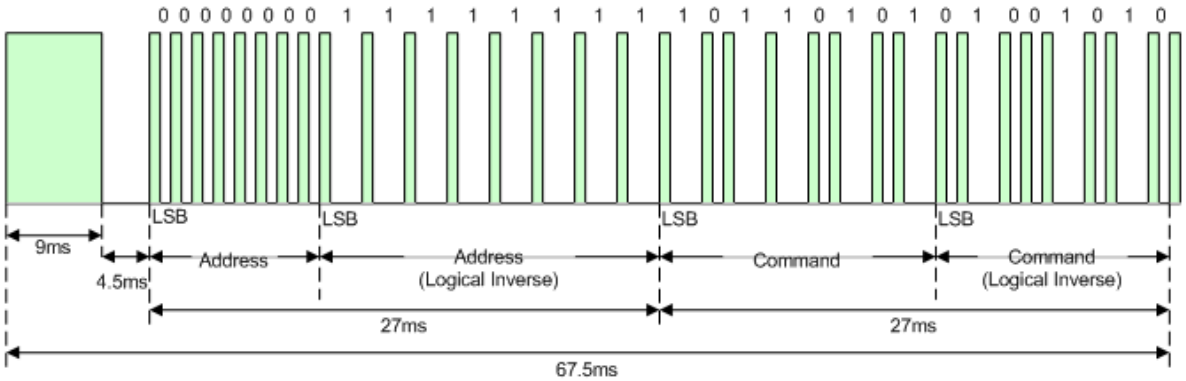


Figure 4: NEC signal Protocol

[1] https://www.vishay.com/docs/80069/circuit.pdf

The NEC protocol allowed us to specify an address defined with 8 bits and a command of 8 bits. The physical signal is sent with 16 bits, the additional 8 bits are used on the inverted address and command for parity checking, which is a method for detecting errors in data communication. This will not be detailed as it has been coded internally in the library and does not require any modification.

In this case we only modified the command to distinguish between signals. In our application modifying the address would not bring notable improvements. The address could, however, be useful depending on the application of the robot.

Unfortunately, the IrRemote library has a limitation in that it was coded in a manner that allowed for only one IR transmitter (or receiver) to be connected at any given time. To get around this issue, we designed a PCB that fits on the Arduino. This PCB contains 4 IR leds as well as 4 NPN transistors and 4 resistors (connected to the base of the transistor). With this method, we can use the same signal pin on the Arduino, and activate or deactivate the IR led that we want to control, by opening or closing the base. The NPN transistors chosen were the BC546 as they were readily available at the electronics labs and performed well as switches.

A specific value of resistor is required at the base to ensure that the transistor works in the saturated region. This allows us to use the transistor as a swich that can be controlled via the Arduino. To calculate the value of the resistor required, the characteristics of the KY-005 were used.

The following calculations were used to determine the value of the resistor:

$$Ibase\ saturated = \frac{Ic}{Hfe} = \frac{20mA}{110} = 0.1818mA$$

$$Rb = \frac{Vf - VCE}{Ibase\ saturated} = \frac{(5 - 0.7)}{0.1818 * 10^{-3}} = 23650\Omega$$

The circuit diagram has been provided in the Annexe. The PCB file is available at the following GitHub address: https://github.com/keanu2311/Robot-Balboa.git

The soldered PCB can be seen in the figure 5 and figure 6 and 7:

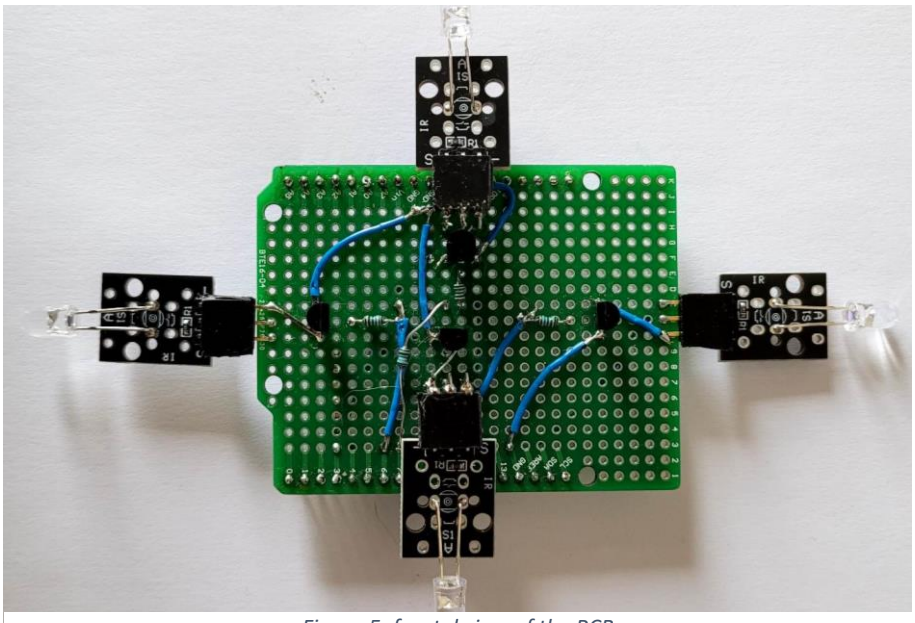


Figure 5: frontal view of the PCB

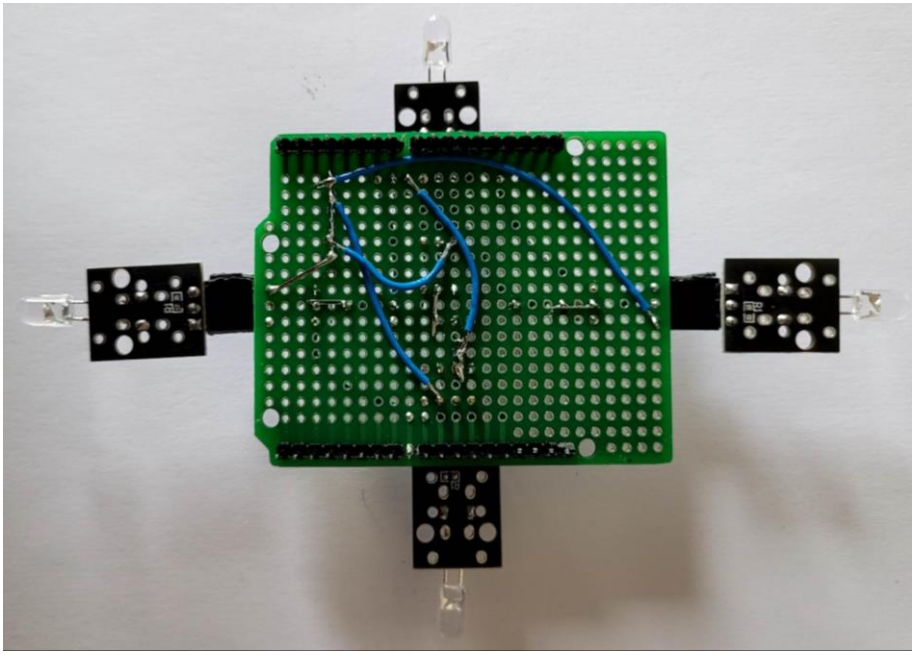


Figure 6: read view of the PCB

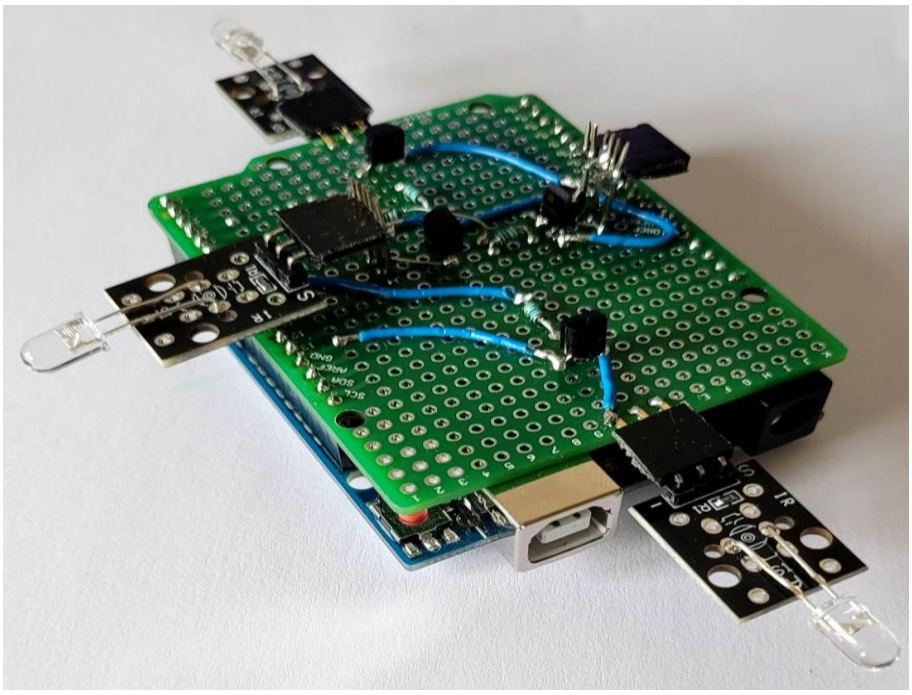


Figure 7: PCB connected to the arduino

4 Wiring diagram

This section will illustrate the pins used to connect the sensors to the robot.

4.1 On the robot

The wiring of the sensors takes two aspects into consideration:

a) The free pins
b) The reserved pins by the system.

The pins on the left and top of the robot are free to use as the datasheet informs us. the Sensors pinouts can be seen in figure 8.

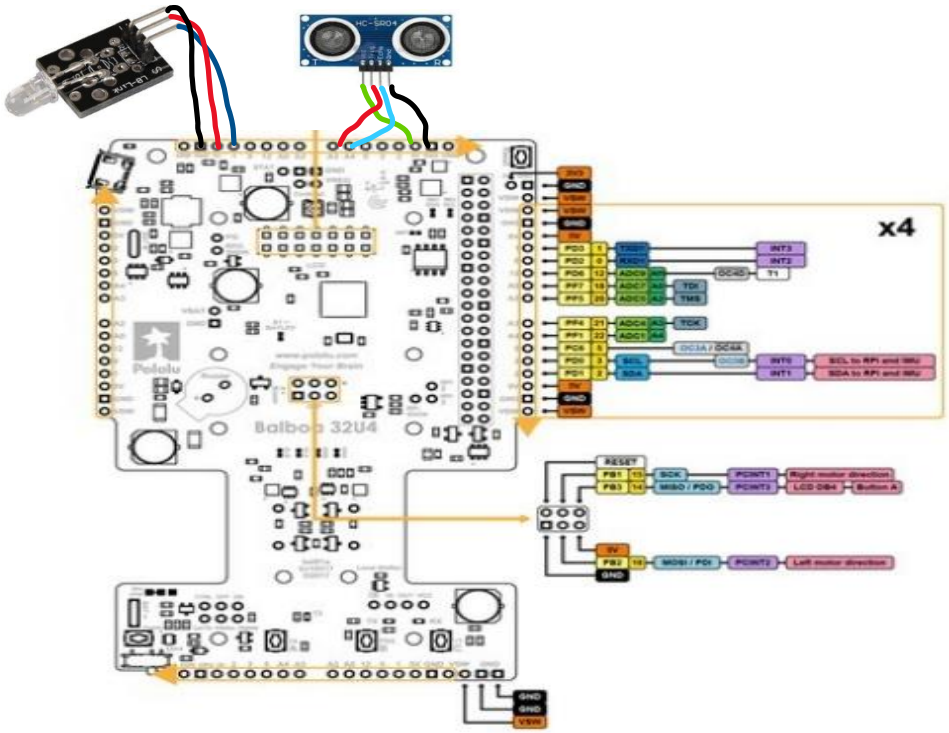


Figure 8: Sensors connections on the Robot

For the IR sensors, the pins used were the 5V (in red), the ground (in black), and the signal on pin 1. For the Ultrasonic sensor, the 5V (in green), the ground (in black), trig on pin A3 and echo on pin A4.

5 Code

In this section we will explain the critical parts of the code to provide a global understanding. For a more detailed understanding the code has been commented which should be sufficient for a more in-depth understanding.

5.1 Algorithm

The following diagram illustrate the algorithm chart that was implemented.

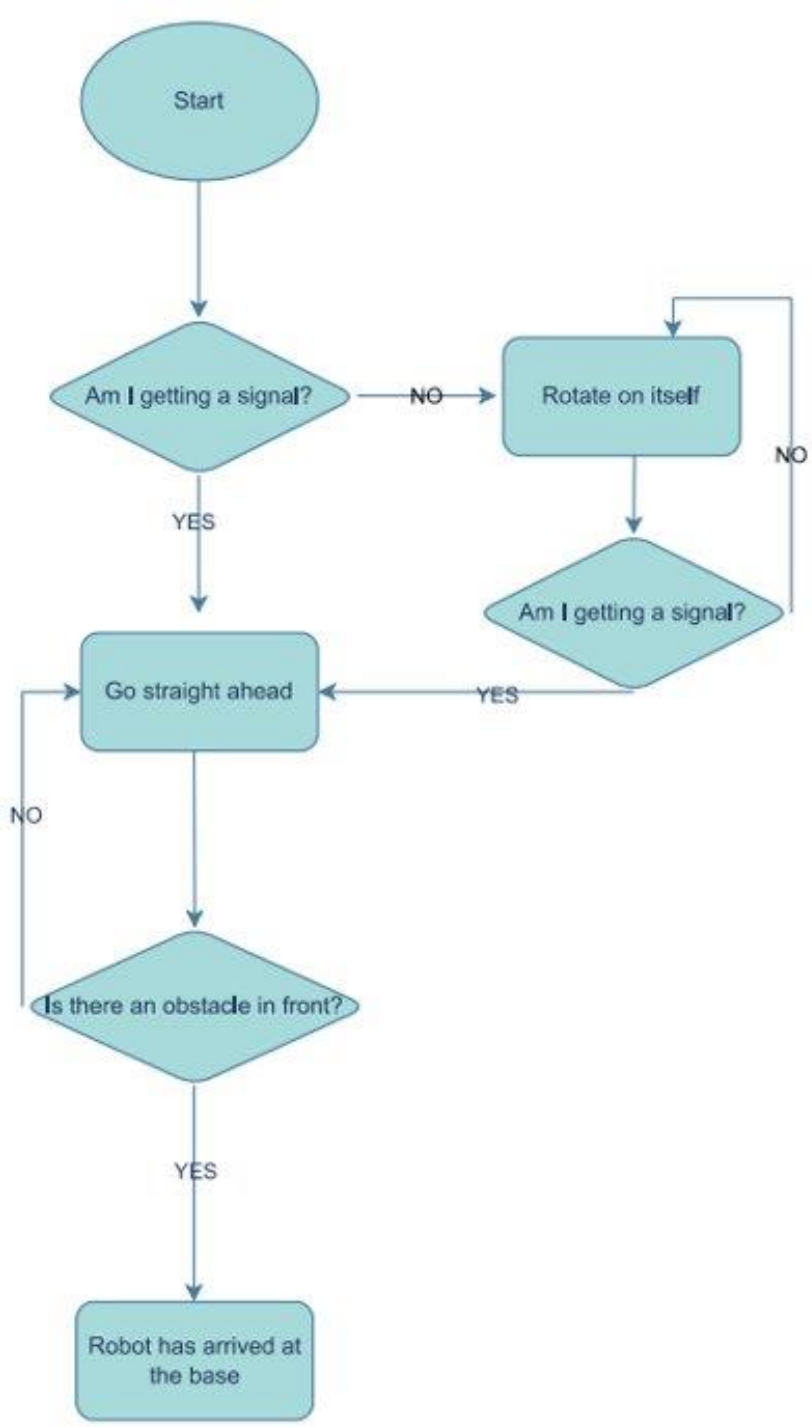


Figure 9: algorithm chart

5.2 Base code

The following code is used to send the IR signal.

```
uint8_t sCommand;
uint8_t sRepeats = 4;

void loop() {
    /

    // Receiver output for the first loop must
    digitalWrite(base1,1);  // activate first ba
    digitalWrite(base2,0);
    digitalWrite(base3,0);
    digitalWrite(base4,0);
    sCommand = 0x34;
    IrSender.sendNEC(0x04, sCommand, sRepeats);
    // send IR data with NEC protocol, OX04 is t
    //aka the number of time it will be sent
    digitalWrite(base1,0);

    delay(10); // delay between each signal is
    sCommand = 0x77;
    digitalWrite(base2,1);
    IrSender.sendNEC(0x07, sCommand, sRepeats);
    digitalWrite(base2,0);
    delay(10);
```

Figure 10: Arduino Base IR Sending code.

The most important part of this code is the sending of the signal. To send a signal, three elements are required:

- The ***address***, in our case 0x04. This address can be changed if need be.
- The ***sCommand*** is the command to distinguish the signal. The values need to be changed for each LED when activating a transistor. Arbitrary values may be assigned as long as they are unique.
- The ***sRepeats*** is the number of times the signal will be repeated. 4 repeats should be sufficient but with trial-and-error different repeats may work better.

Those three elements need to be added using the method *". **sendNEC***".

After the signal is sent, the transistor needs to switch off by deactivating the current transistor base and activating the next one, using ***digitalWrite().***

5.3 Robot code

On the Balbao, the two main methods needed to use the Sensors are called ***IR_detectionV2()*** and ***UltraSound_sensor ().***

1) The function ***IR_detectionV2()*** has been implemented using the following code extract:

```
void IR_detectionV2()
// function used to decode the incomming IR si
{
  IRsignal1 = false;
  IRsignal2 = false;
  IRsignal3 = false;
  IRsignal4 = false;
  detected_IR = false;
 if (IrReceiver.decode())  // if signal has be
  {

    IrReceiver.resume(); // Enable receiving o
    switch (IrReceiver.decodedIRData.command)
     {
        case 0x34:
            IRsignal1 = true;
            detected_IR = true;
            break;
        case 0x77:
            IRsignal2 = true;
            detected_IR = true;
            break;
```

Figure 11: IR detection function

When entering this section of the code, we ensure that each signal is set false so that the previous command stored in memory does not affect the new incoming command.

We enter the ***If statement*** only if a signal has been detected. Each command is than compared in a switch case. If the signal matches a particular case, the IRsignal is set to true. This IRsignal variable will be used in the void loop.

2) The function ***UltraSound_sensor ()*** is the comprised of the following code:

```
void UltraSound_sensor()

{
                                          //fonctinc
    dist=distanceSensor1.measureDistanceCm();


    if ((dist>15)) // if distance is supperieu
    {
      forward = true;

    }
    else
    {       // stop robot if < 15 cm
      forward = false;
    }

    count_millis=millis();  // start counter t
    in_function = true; // to know if we passe
}
```

Figure 12: UltraSound_sensor function

The variable ***dist*** stores the measured distance to an object. As long as the distance is superior to 15 cm, the robot will continue to move forward. The value of 15 was chosen as a security distance in order to prevent the robot slamming into the Arduino base. The variable ***forward*** is set to true and will be used later in the loop function, to give the command to the robot to continue to move forward.

3) The ***main loop ()*** function comprises of:

```
void loop()
{

  balanceUpdate();


  if (in_function == false)
    {
     UltraSound_sensor();
    }

  if ((IRsignal1 || IRsignal2 || IRsignal3 || IRsignal4) == false)
    {
      IR_detectionV2();
    }


    if (detected_IR == true)
      {
        leftSpeed = -10;      // speed of both motors are put as the s
        rightSpeed = -10;
        balanceDrive(leftSpeed, rightSpeed);  // pass in leftSpeed ar
        if ((millis()-count_millis) > delay_forward) // if the elaspe
          {
            in_function = false;
            IR_detectionV2();
          }
        if (forward == false) // if we detected an obstacle, the robc
        {
          leftSpeed = false;
          rightSpeed = false;
          balanceDrive(leftSpeed, rightSpeed);

        }
      }
```

Figure 13: main loop function

The first essential function called on entering the loop will balance the robot. Once it is balanced, the functions ***IR_detectionV2***() and ***UltraSound_sensor*** () are called to read the incoming values.

If an infra-red signal has been detected, the variable ***detected_IR*** will be true, and the robot moves forward. It will continue to move forward until the next reading of the IR is completed. If the elapsed time since the start of the code and the time spent within the ***UltraSound_sensor*** () function, is greater than 1000ms (***delay_forward***) then we reread a value. This done so as not exceed the sampling time of 10 ms. Without this condition the robot would read too many values and would fall over. The ***delay_forward*** value was chosen through trial and error.

6 Limitations & improvements

In this section the limitation of the robot will be discussed, as well as possible avenues of improvement.

6.1 Base limitations

The Base has 3 main limitations:

First limitation:

To start off, the positions of the IR LEDs create a few blind spots. As a result, the robot won't detect any signals in those particular areas. To solve this, the robot rotates on itself until it finds a signal. This is not the optimal implementation.

There are 2 solutions:

1) 4 additional IR LEDs could be added which would remove most of the blind spots. The PCB would however need to be modified to incorporate the additional LEDS, transistor and resistors needed.
2) The algorithm could be improved, so that the robot moves around a room until it detects a signal. This solution isn't ideal either, as it would be dependent on randomly detecting a signal. A combination of both solutions could prove to be most effective.

Second limitation:

The second limitation are the heights of the IR transmitters. If the base is positioned too low or too high, there is a risk that the robot might not detect the signal or only partially detect the signal.

To solve this issue, more receivers could be installed on the robot at various heights, such as at wheel height, at central of mass height and at the highest point on the robot. The algorithm would need to be updated to include these extra sensors, as a well require the inclusion of another PCB to house the sensors.

Third limitation:

Finally, the PCB was not professionally produced. The PCB was soldered and connected with small jumper wires in the electronics lab. If the soldering was not done adequately, there could be a risk of poor or broken connections, or in the worst-case scenario, a short circuit that would destroy the transistors and the LEDS.

The simplest solution would be to use the PCB that we designed and have a professional company print out the circuit board.

6.2 Robot Sensors

First limitation:

The use of the ultrasonic sensor is not ideal to detect if the robot as reached the base. If the base does not have an obstacle in view, it would not be possible to determine when the robot needs to stop. Ideally, this Ultrasonic sensor should be used to detected objects that are in-between the robot and the base, so that the robot can avoid the obstacles (with the need to implement an obstacle avoidance algorithm to go around it).

One possible solution is to use passive RFID sensors on the base and the robot. Such sensors send and receive signal at very close range. The combination of the IR signal for direction, and the short-range signal from the RFID could be used to more accurately detect when the base has been reached.

Second limitation:

The robot only has one IR receiver. This is problematic because the robot does not have any spatial awareness as to where it is in the room, or from which direction it receives a signal (left, right or front of the robot). Because it only has one IR receiver, the only solution to locate the signal is to rotate on itself continuously. As soon as the signal is lost it needs to rotate once more, until it detects a signal, which is not the optimal implementation.

To solve this problem, 4 additional sensors should be added to the robot. One placed on the left side of the robot, one on the right and two more at different heights in front of the robot. By doing this, the robot would know in which direction it needs to rotate to when a signal is detected, instead of completing a full rotation. This solution would increase the speed at which the robot would be able to return to the base, because if it loses the signal in front, but manages to detect it on either side, only a small course correction would be needed, by rotating slightly to the appropriate side.

To implement this solution, a PCB is required with more transistors as the IRremote library does not allow for more than one Receiver to be connected at any given time (This uses the same methodology as the IR transmitters).

7 Conclusion

In conclusion, the main objective of returning to the base was achieved. As this was a first prototype for the project, we developed simplified algorithms that would require more work. With additional time and resources, the robot could be improved to be more efficient in returning to the base.

Unfortunately, due to the time constraints and the complexity of the project, the temperature and smoke sensors could not be implemented in time. It is recommended that the limitations discussed be overcome before implementing the temperature and smoke sensors.

8 Bibliography

IRremote Library by Ken shirrif:

https://github.com/Arduino-IRremote/Arduino-IRremote

Ir receiver's datasheet by Vishay:

https://www.vishay.com/docs/80069/circuit.pdf

IR protocols documentation by Holtek:

https://www.bestmodulescorp.com/amfile/file/download/file_id/1939/product_id/797/

NEC transmissions Protocol by Altium:

https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol

Radio Spectrum by center for democracy & technology:

https://cdt.org/insights/techsplanations-part-6-mobile-connectivity-an-incomplete-explanation-of-the-radio-spectrum/#:~:text=In%20short%2C%20longer%20wavelengths%20travel,wavelength%20and%20frequency%20are%20related.

Transistor Datasheet:

https://octopart.com/bc546-on+semiconductor-7765551?gclid=CjwKCAjwge2iBhBBEiwAfXDBR690EPt52bX6D178E-o2uUsvdJMkD3sRQ0ce0-Qp4Sof-D__AMtt8RoCdj4QAvD_BwE

KY-022 datasheet:

https://www.epitran.it/ebayDrive/datasheet/45.pdf

KY-005 datasheet:

https://datasheetspdf.com/pdf/1402020/Joy-IT/KY-005/1

9 Annexe

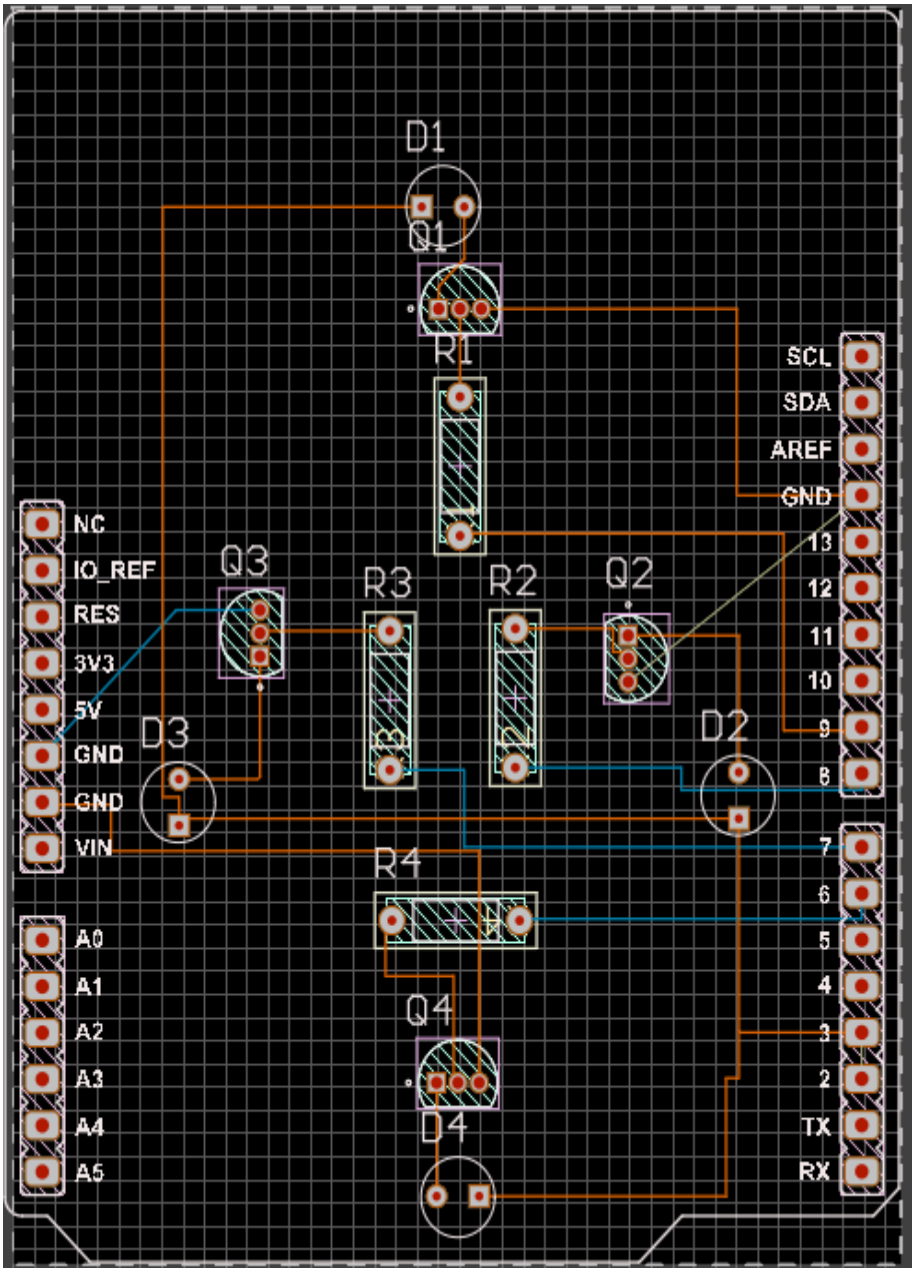


Figure 14: PCB layout

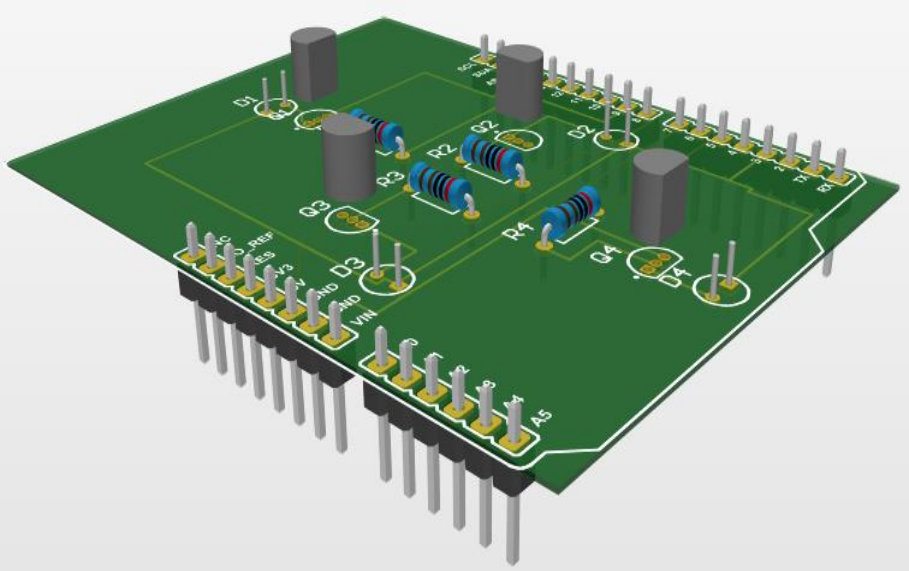
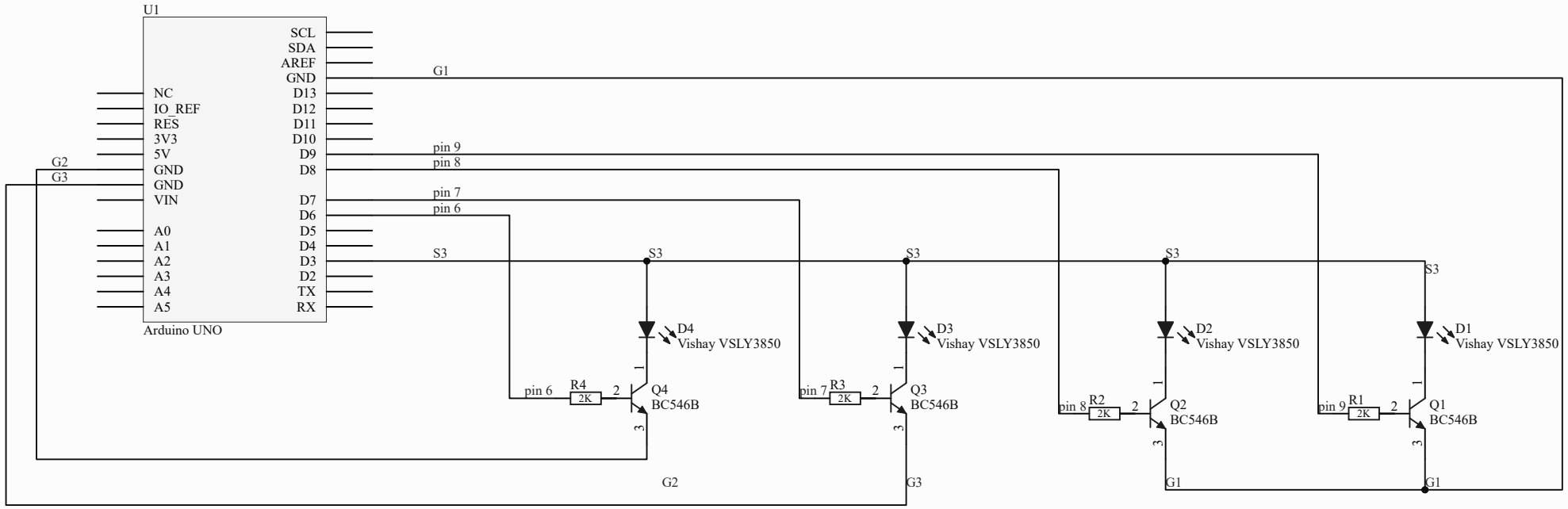


Figure 15: 3D layout of the PCB



Title IR sensor Schematic		
Size A4	Number	Revision
Date: 5/10/2023		Sheet of
File: balbo_arduino_shield.SchDoc		Drawn By: Buch Keanu