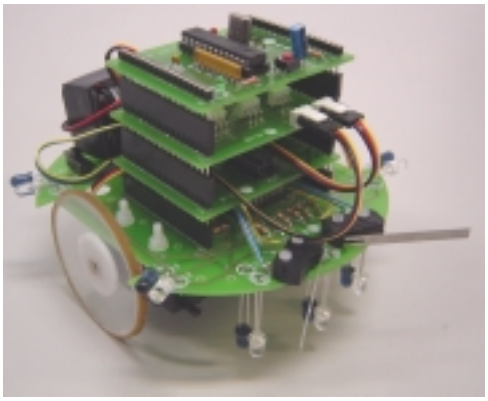
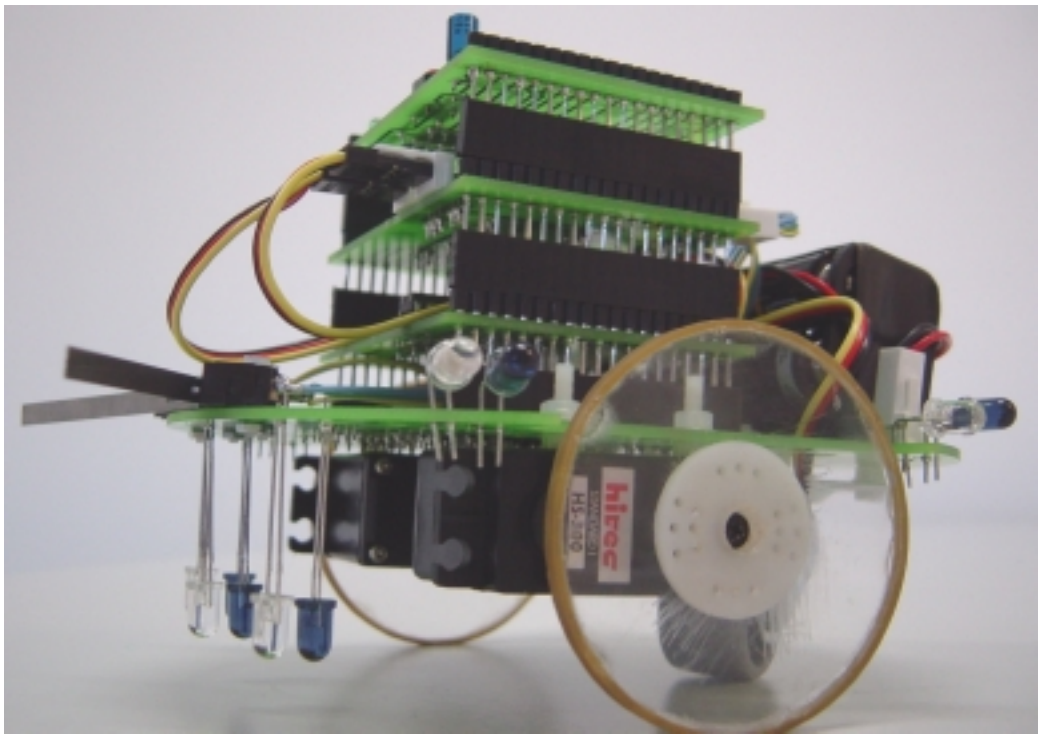
IdMind – Engenharia de Sistemas, Lda
E-mail: info@idmind.pt
http://www.idmind.pt



Circular Robot II



April 2003

Preface

Congratulations! You have just acquired a kit with IdMind' quality!

At IdMind we see the information technologies as a way of efficient learning. Those capabilities are enlarged when complemented by practical applications. We firmly believe that **learning while doing** is an efficient way to transmit knowledge.

The inherent multidisciplinarity of Robotics gives us an ideal tool for the understanding of different aspects of science and technology, in particular the ones involving Electronics, Mechanics and Computer Science.

The Circular Robot kit is supported on a circular platform with the dimensions of a CD. Its main feature is modularity: the robot is capable to accommodate different sensor types. The execution of a task is made possible by programming a microcontroller. The robots can be programmed using a PC, through a high-level visual language, where the programmer just needs to implement a flowchart for the desired robot behaviour inserting and linking blocks, allowing the robot to perform a set of different tasks only limited by the programmer's imagination.

This manual permits an ample understanding of the different inherent details to the robot's construction. However, for any additional clarification, we are available at Internet:

Email: info@idmind.pt

http://www.idmind.pt

Table of Contents

1 Introduction

Robots have been in one's imagination throughout the times. Shortly ago, everyone's thoughts were that robots were a huge complex system, very expensive, consisting of an amalgam of electrical wires and complex computerized systems.

However, just in the few last years, the achieved advances in the microcontrollers technology, along with the drastic reduction of dimension and cost in opposite with the amazing increase of potentialities, made possible the accomplishment, in a simple way, of a set of robotic systems capable to develop autonomously several tasks.

It must be stressed that the robot construction is not just its programming. Its conception involves a set of inherent knowledge in mechanics, electronics and computer science. This complex process gives us an extremely involving challenge.

In this manual are detailed all the particular aspects of the small robot construction and it is supported by an electronic kit developed by the authors of this project. The user will acquire, alongside with the construction process, the perception of the different stages of the robot conception, beginning with the components soldering and finishing with the microcontroller programming.

1.1 Robot' Functional Structure

The purpose of the robot' functional structure is the reply for these three basic problems:

Where am I? Where am I going? How can I go?

When a robot moves in a specific environment, it uses its sensors in order to both locate itself and to identify its targets. It uses the actuators in order to dislocate itself or to manipulate any object. At last, it uses its microcontroller in order to coordinate all the actions of perception and actuation.

Figure 1.1 synthetizes the way it works:

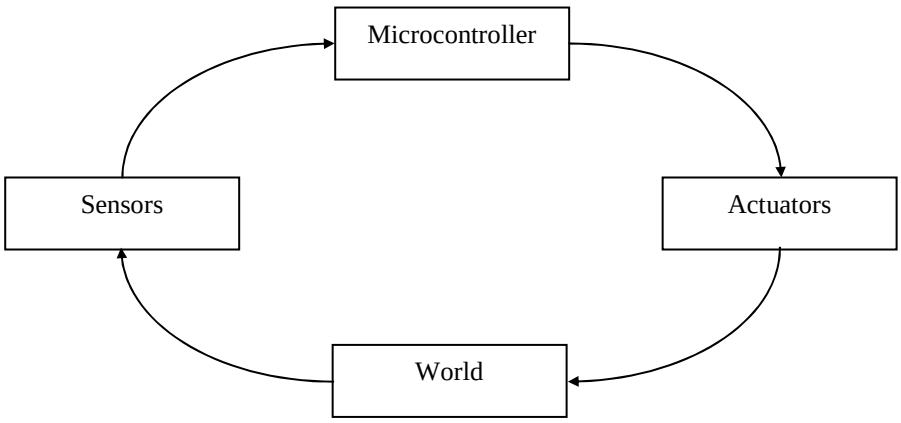


Figure 1.1 – Robot Functioning Cycle

2 Robot Construction – Some factors to bear in mind

Since this manual is focused on the details about the conception of the electronic components and the robot programming, in this section we pinpoint the details one must bear in mind when constructing a robot.

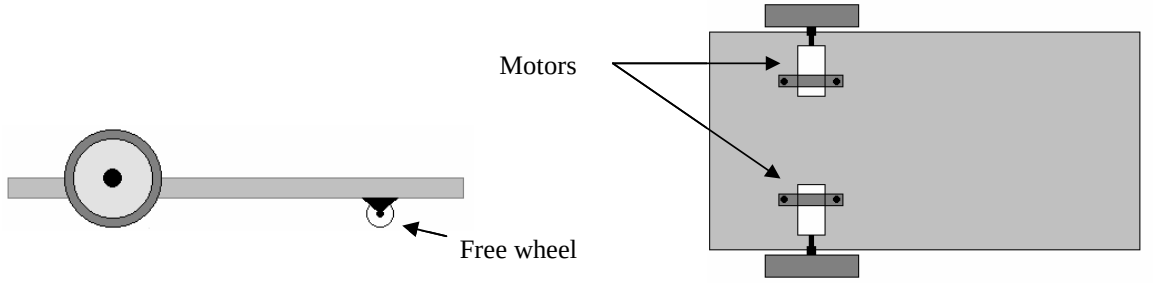
2.1 Platform Construction

The platform may assume different shapes, however, in its conception, one must be aware about the following factors:

- ✓ Simplicity – minimize the number of mobile parts and the complexity of robot;
- ✓ Robustness – shock resistor;
- ✓ Modularity – the robot must be composed by a set of connected modules, in such a way that if one of the modules needs to be removed there is no necessity the removal of the remaining.

Different materials could be used, but the most common are the PVC, the stainless steel, the aluminium, the plywood and the acrylic.

Figure 2.1 shows a usually used structure for the robot. It includes two independent motors (differential traction) and a free wheel to keep the balance.



a) Lateral view b) Top view

Figure 2.1 - Robot with differential traction

2.2 How to choose the Motors

In order to choose the adequate motor for a specific robotic application, some factors should be bear in mind:

- ✓ Size and shape;
- ✓ Mass;
- ✓ Available Binary;
- ✓ Maximum speed (without commutation failure, nor damaging the motors);
- ✓ Maximum Current/Binary (without superheating, with waste dissipation).

Among all the different motors one can find in the market – continuous current, step by step or alternating current - the continuous current motors, also known as DC (Direct Current) motors, are the most divulged, due to its better relation power/volume and diversity.

2.2.1 Binary and Speed Adjustment

Usually the motor is not prepared to be directly linked to a wheel, sometimes due to an excessive speed or, other times, due to an insufficient binary. In order to adapt the motor to one's needs, it must be used a system of reduction (toothed-wheels or pulleys) in order to increase the binary and decrease its main shaft speed.

2.2.2 DC Motor Speed Control

A motor speed can be controlled through its input voltage (current). However, to prevent the components superheating, instead of using a continuous signal is used a PWM (Pulse Width Modulation), being the impulses width the controller of the sent voltage which control itself the rotation speed. The motor speed varies proportionally to the area underneath the positive portion of each period (figure 2.2).

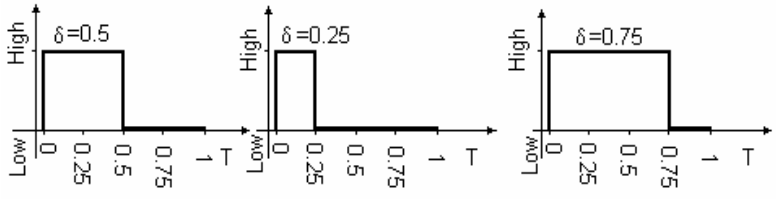


Figure 2.2 –PWM Signal

Since several microcontrollers hold this kind of channel, it becomes very easy to use it. On the other hand, the thermal losses in the components are reduced, because not always the voltage is applied.

2.2.3 Direction Control

The inversion of the motors rotation can be obtained reverting the applied voltage. The classic solution for this situation is the use of a diagram H-Bridge type, such as it is shown in figure 2.3. The PWM signal could be applied in the *Enable* terminal.

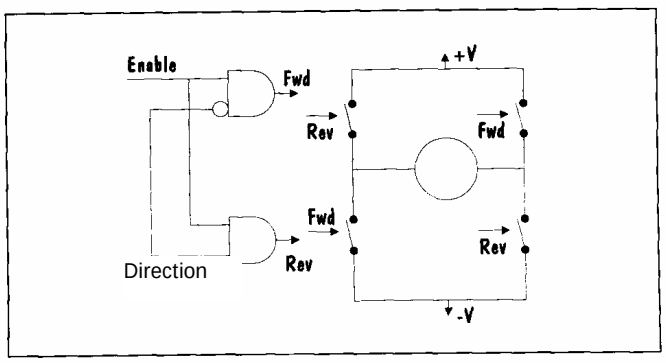


Figure 2.3 – H - Bridge

3 Kit' Enclosed Components

The Circular Robot electronic *Kit* includes:

- Controller board with PIC (ID-PIC002);
- Digital interface board (ID-DS002);
- Analog interface board (ID-AS002);
- Serial programming board (ID-PROG002);
- Board as a Base with interface for the IR sensors (ID-BASEIR002);
- 2 Servomotors HITEC HS300;
- RS232 communication cable (9 pin plug);
- Other small components.

3.1 General Description

This kit includes all the electronic components necessary to construct a small robot.

This kit was conceived to be of easy assembling and programming, allowing its use by everyone's interested in the areas of robotics and automation.

The robot to construct will develop everyone's capacity to:

1) Identify and build electronic circuits;
2) Understand its different parts:
   - sensors;
   - actuators;
   - microcontrollers;
3) Develop one's own program, taking the robot to execute a set of different tasks.

3.1.1.1 Sensors

It includes seven pairs of infrared emitters/receivers which allow the detection of, e.g., the existence of a painted line on the floor or even an obstacle. It still includes two contact switches, which allow the detection of an obstacle collision.

3.1.1.2 Actuators

A pair of angular variation servomotors is modified to have angular speed variation.

3.1.1.3 Microcontroller

The Microcontroller is the robot's "brain". Inside it, the information, coming from the sensors, is treated and it is taken all the movement decisions.

3.2 Functioning

The ID-PIC002 board is a printed circuit board where the PIC16F876 microcontroller must be placed. This board allows the sensors and actuators connection, using two side bars, witch allow the access to all the pins of input/output ports (E/S).

Included in the kit are two other boards which allow the access to the several pins of input/output microcontroller. One of them is for the digital pins (ID-DS002) and the other one to the analog pins (ID-AS002). In this last one is possible the use of only one analog entrance in order to read the height multiplex analog entrances.

In the kit is also included an Infra-red module (ID-BASEIR002) to allow the robot to follow a line white/black painted on the floor or detect obstacles. This module uses 7 pairs of infra-red sensors, and it is the robot's base.

Since the used processor does not allow the simultaneous reading of the 7 receivers, this board was designed to permit one and only one of the emitters to work in each instant of time. This method allows an energy saving.

The basic principle of its functioning is very simple: in the emitter it uses only one resistor of 100 Ω, and an infra-red LED emitter. By the side of the receiver it is enough to bind a resistor between the phototransistor emitter and the mass, being the collector connected to 5V and the reading of the signal made by the phototransistor emitter.

The connection of the output analog sensors is very simple, being enough to connect the sensors output bolt to the available input analog signal of this robot, more concretely in the analog interface board ID-AS002. In this board there are 4 analog inputs, and the channels 1 to 4 of the *PortA*, are available in the terminals PA1, PA2, PA3 and PA5. The terminal PA4 is a digital input/output.

The analog interface board ID-DS002 offers 11 digital input/output that can be used to increase the number of sensors/actuators. In this kit are included 2 *micro-switches* which can be used in one of those inputs. As actuator, we choose two modified servo motors that are actuated by the existing PWM input in the microcontroller. The use of the servo motors is an advantage since they already have connected to their main shaft a reduction box and also have in its interior an electronic circuit which affects the H-Bridge. Figure 3.1 resumes the robot functioning.

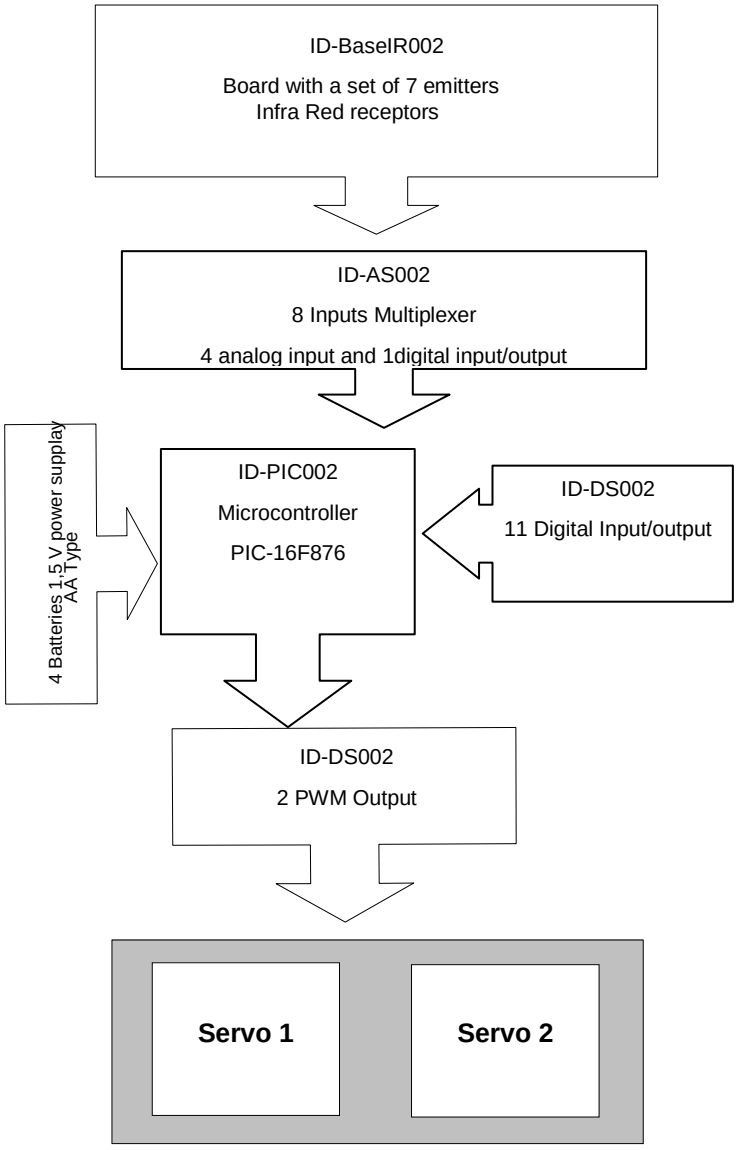


Figure 3.1 –*Hardware* architecture

4 Components Assembly

4.1 Basic Principles of Soldering

The success of the kit assembly depends, in a great way, on the components soldering in the printed circuit boards (figure 4.1). The fulfilment of some basic rules in this process becomes thus necessary:

- When possible, should be used a fine tip soldering-iron and with a not superior 25 W power. This kind of product is found in an electronic specialized store;
- During its use, always keep clean the tip of the soldering-iron using a wet sponge.
- It should never be used a dirty or sandpapered tip, being in this case, preferable its substitution;
- It should be used a 0,7 mm solder;
- Before soldering, always clean the component pins with a fine sandpaper;
- Hold the soldering-iron, place its tip on the component terminal, and apply the solder wire (figure 4.1). This process should not take more than 5 seconds (not carrying out this rule can lead to the destruction of the component);
- After accomplished the soldering, its surface should present a metallic brightness, being this the ideal moment to cut the components extremities;
- If you need to remove the solder please use a solder remover;
- While soldering a sensitive component, one must hold it with a clamp in order to dissipate the heat. Other way, the component can be damaged
- It should never be used excessive solder in order to avoid a short circuit between tracks. (figure 4.2)

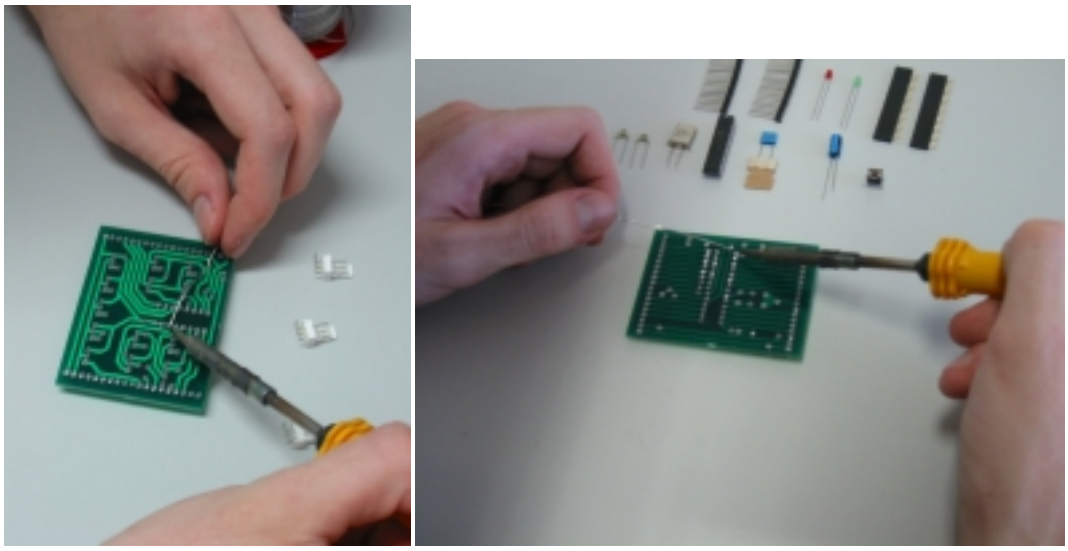


Figure 4.1 – Components soldering

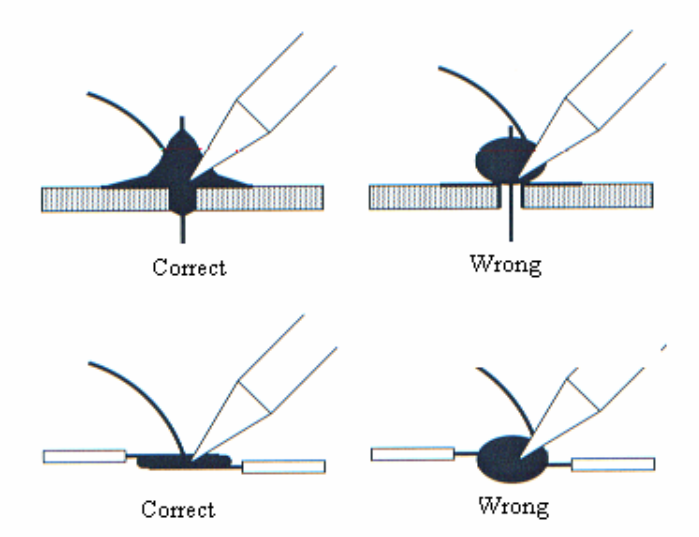


Figure 4.2 – Soldering Methods

4.2 Components

An electric circuit is composed of several components with different shapes, sizes and codes, permitting its distinction. In a circuit construction the user should know how to identify the different components.

Be aware that some of the components are polarized and its assembly must be made in only one way. A deficient components assembly can lead to a deficient functioning or even to the destruction of the electric circuit.

This section goal is to familiarize the user with the different components existing in a simple electric circuit.

4.2.1 Resistors

Usually the resistors are small cylinders with some coloured stripes. The resistors included in this kit are of ¼ *Watt*, which corresponds to a low power, but sufficient for the robot good performance. The colour code is the standard, consisting of 4 coloured stripes around the cylinder. The first two stripes are the mantissa and the third one the exponent.

A resistor can be identified by forming a number whose cipher ten corresponds to the first coloured stripe, and the unity cipher corresponds to the second coloured stripe (Table 1- colour correspondence). The third coloured strip corresponds to the exponent of a base 10 power, which must be multiplied the number gotten by the first two stripes.

The fourth stripe represents the resistor tolerance. If this stripe is silver-plated, the resistor has a tolerance of 10%; if it is golden, then the tolerance is 5%. The tolerance represents the interval of error of the nominal value of the resistor supplied for the manufacturer. Example: for a resistor of 100Ω, with a 5% tolerance, the resistor real value could be between 95 and 105Ω.

Colour	Stripe Value	Multiplicative Factor
Black	0	1

Brown	1	10
Red	2	100
Orange	3	1000
Yellow	4	10000
Green	5	100000
Blue	6	1000000
Violet	7	
Grey	8	
White	9	

Table 4.1 – Resistor Colour Code

For example, a resistor having the fallowing colour sequence: orange, white, and red,

Orange = 3
White = 9 $\Rightarrow 39$ $\Rightarrow 39 \times 10^2 = 3900\ \Omega$
Red = 2 $\Rightarrow 10^2$

The resistor has the value of 3900Ω.

4.2.2 *Pack* of Resistors

Another kind of resistors can be found in an electric circuit. They are not unitarian, they are grouped in sets that vary from 3 to 9 resistors (figure 4.2 represents one pack of 7 resistors).

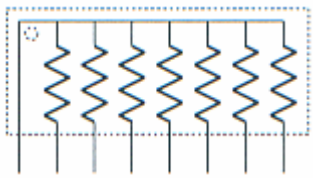


Figure 4.2 – Pack of resistors

In the kit, the used set has 8 resistors plus a common bolt (identified with a dot). Notice that this component is polarized, being necessary to verify if it is correctly placed before soldering.

4.2.3 Capacitors

There are several types of capacitors offered for sale: monolithic, electrolytic and tantalum.

4.2.3.1.1 Monolithic

They are made of Polyester or ceramics, and usually they are small and have low capacity (pF or nF). This kind of capacitors is never polarized.

4.2.3.1.2 Electrolytic

This kind of capacitors usually presents a cylindrical shape, coated by a plastic encapsulation, and they present high capacity values (> 1 μF). This type of capacitor is polarized and its physical dimension increases with the increase of its capacity.

4.2.3.1.3 Tantalum

These are highly compact capacitors, and they have the shape of a drop. Their capacity values are in the same order as the previous type, but the advantage of their use is its reliability being the disadvantage the higher cost. These capacitors are always polarized.

Note: The polarized capacitors could explode when its polarity is not respected. For this reason, the user must respect the capacitor polarity on the occasion of its assembly. It should be used the indication of (+) and/or (-) of its package. There are several ways to indicate the value of the capacitor, but the usual, for capacitors larger that 1μF is that the value is written in the component. In some cases the μ, works as a decimal point, and the 4,7 capacitor is represented by 4,7μF. For small capacitors the values are represented by picofarads (1000000pF=1μF). Another way to represent it is, for example 473, where the procedure is equal to the one used for the resistors. This means: 47x103pF=47000pF=0,047μF.

4.2.4 Diodes and Leds

The diodes and the leds (Light Emitting Diodes) have two pins: the anode and the cathode. The anode must be connected to a positive voltage terminal, in relation to the cathode, in order to allow the flow of current. When the polarity is not respected the component does not allow the passage of current. Figure 4.3 shows how to identify the diodes and the leds pins.

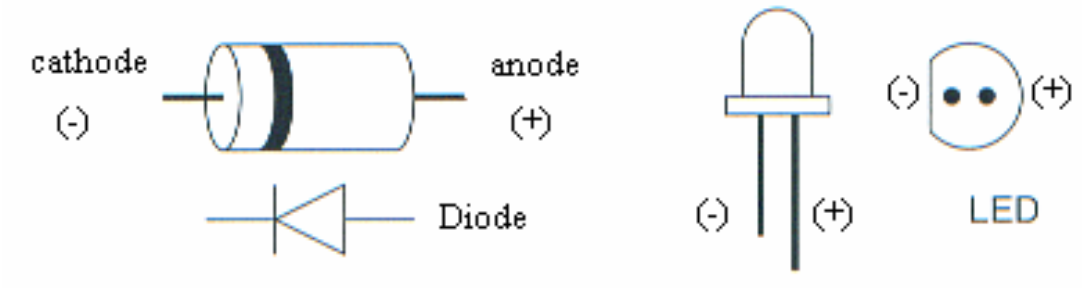


Figure 4.3 – Diodes and *leds* polarity

Usually the diodes are small cylinders with a small stripe around it, which allows the cathode identification. The leds are diodes that emit light when a current flows by its terminals. Its polarity identification is made by the verification of a longer “leg” (anode (+)), or by its side, where the led circumference has a small flattened part (cathode (-)).

4.2.5 Integrated Circuits

The integrated circuits (IC) are components internally composed of complex circuits, and they can be found for sale in different shapes and sizes. Usually, in the assembly of manual circuits, it is used the DIPs (Dual-Inline Packages).

The integrated circuits must be assembled bewaring of its numeration, which is not written on its surface. However, it can be found two marks showing the bolt number 1. These two marks can be observed as the re-entrances in the chip - the first one is a small half-moon located in one of the boltless sides of the chip, which indicates that the bolt 1 is located on its right (when the half-moon faces the user), and the second mark is a small circle located over the bolt number 1 (figure 4.4). A deficient assembly of this component usually leads to its destruction.

Figure 4.4 – Integrated Circuit Shape -Type *DIP*

4.3 Construction of the Robot Boards

In this section is explained the assembly of the set of boards that constitute the robot. Figure 4.5 shows the set of printed circuit boards included in this electronic kit.

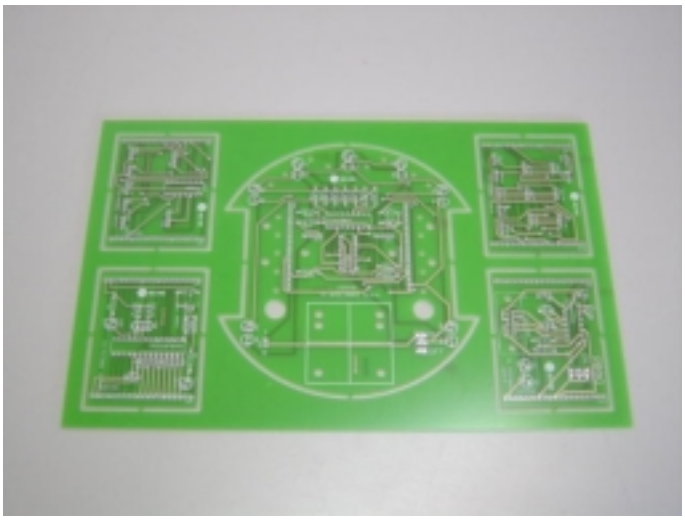


Figure 4.5 – Set of boards used in the robot construction

4.3.1 Construction of the Control Board ID-PIC002

Start separating the ID-PIC002 board (fig. 4.6). For such, cut the small connections that join it to the set of boards and, if necessary, use a fine sandpaper. Identify the components used in this board.

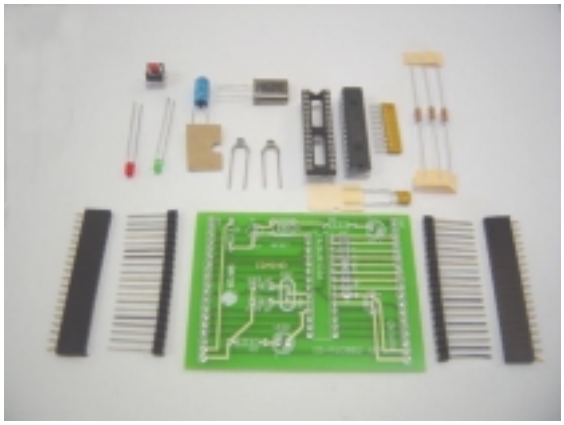


Figure 4.6 – Components used for the construction of the board ID-PIC002

Begin the construction of the microcontroller board by soldering the three 3,3KW resistors (section 4.2.1) in the marked places on the board. After doing it, solde the 9 pins resistor (section 4.2.2) bewaring that the dot on the resistor corresponds to the square shaped mark.

In order to facilitate the orientation of this component, a white circular mark was drawn. This white circular mark must coincide with the square or circular shaped mark of the resistor.

The 28 pins socket of the microcontroller must now be soldered. Beware of the shape drawn on the board, must guide the socket in order to coincide both half-moons (section 4.2.5).

Solde the two leds bewaring of its polarization (section 4.2.4, green led on “LED 1”, and red led on “LED 2”). Solde the two 15 pF capacitors, and the 100 nF capacitor. Solde the electrolytic capacitor bewaring of the indicated polarization (section 4.2.3). Solde the pressure button on “S1”, and the crystal on “CRZ”. Finally, sold the two 19 pins male bars, and insert in them 2 female bars. Place the PIC16F876 microcontroller in the respective socket.

Figure 4.5 shows the microcontroller board after the components assembly. Notice that this board has two 19 pins male and female bars. The male bars are soldered on the board and the female are inserted in the male bars.

Front side

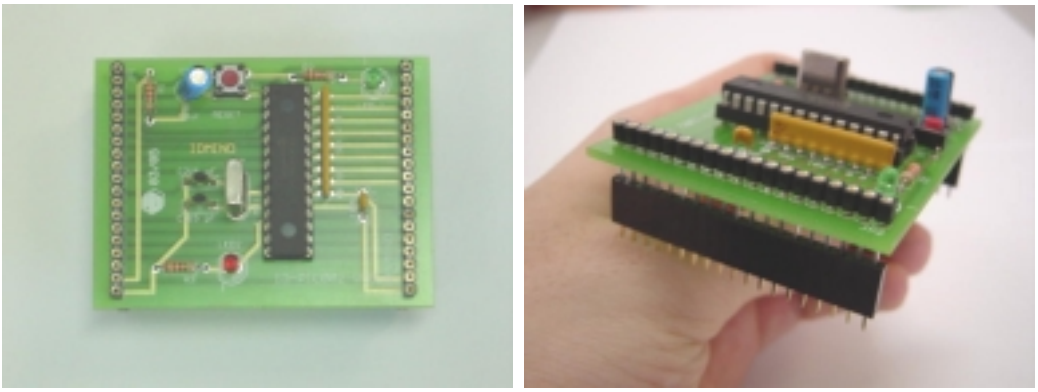


Figure 4.7 – Microcontroller Board

4.3.2 Construction of the Digital Board ID-DS002

Separate the ID-DS002 board (fig. 4.8). For such, cut the small connections that join it to the set of boards and, if necessary, use a fine sandpaper. Identify the components used in this board.

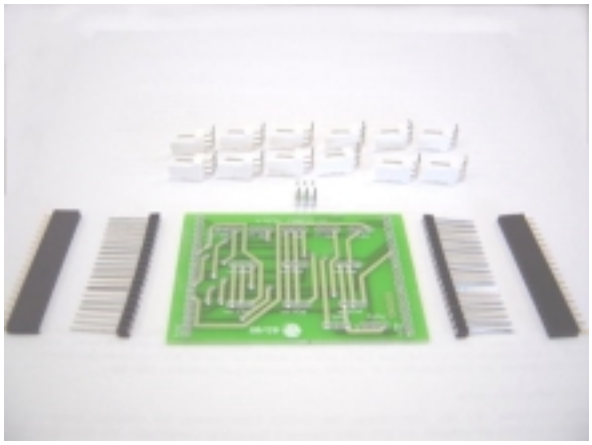


Figure 4.8 – Components used for the construction of the board ID-DS002

Begin the construction of the digital board by soldering the several terminals of three pins (with the angle of 90º), bewaring of the two PWM terminals will have to make a 180º angle with the first ones (fig. 4.9). Solde the two 19 pins male bars on this board, and insert in these the 2 female bars.

Figure 4.9 shows a digital board assembled. All the bars of the PortB and some of the bits of the PortC are accessible. Each sensor is connected to one of the 3 pins terminals. In figure are pointed, on each terminal, the corresponding pins to the mass signal "-". The central bolt supplies a signal of 5 V and the other bolt corresponds to the signal bolt.

The same figure shows that the linking terminals C1 and C2 (PWM2, PWM1) make an angle of 180º with the remaining terminals, being the connections angle also of 180º. As the controller board, this board has the two 19 pins male bars soldered to the board, and the other bars (female) inserted in the first ones.

Front side

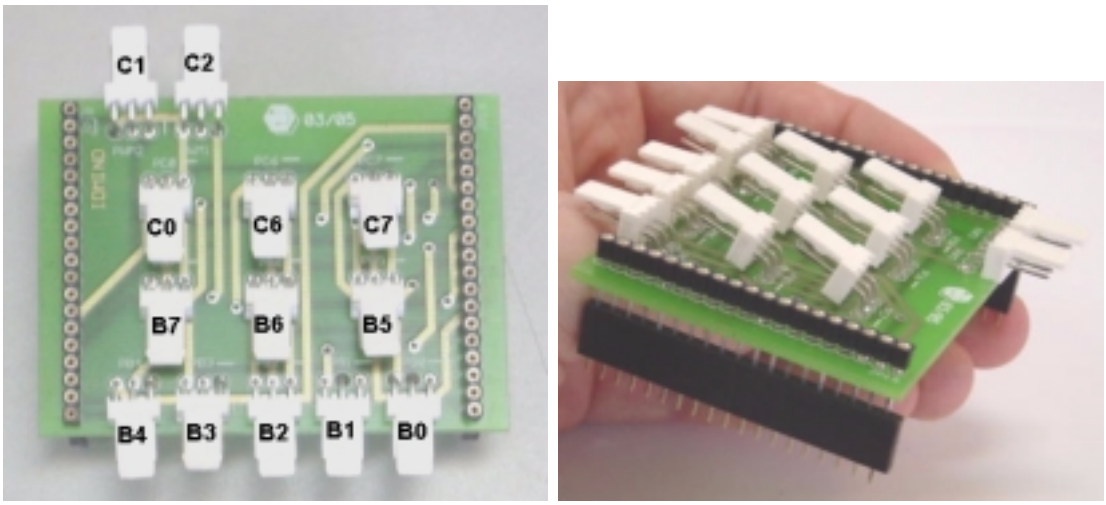


Figure 4.9 – Digital Interface Board

4.3.3 Construction of the Analog Interface Board ID-AS002

Separate the ID-AS002 board (fig. 4.10). For such, cut the small connections that join it to the set of boards and, if necessary, use a fine sandpaper. Identify the components used in this board

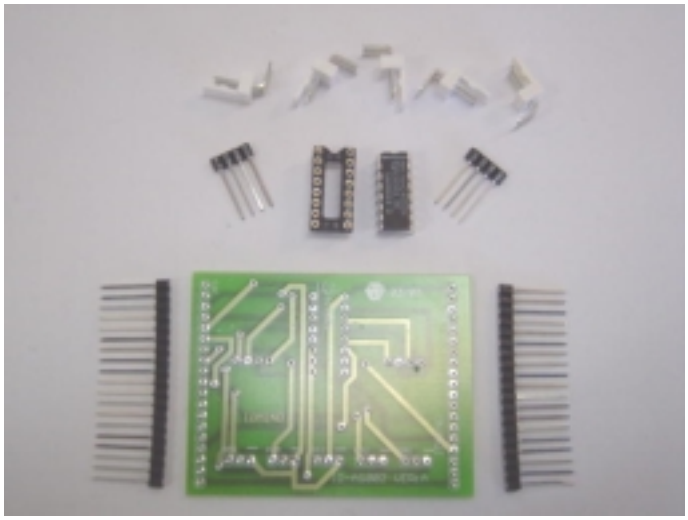


Figure 4.10 – Components used in the construction of the ID-AS002 board

Begin the construction of the analog board by soldering the several terminals of three pins (with the angle of 90º), in the places indicated for PA1, PA2, PA3, PA4 and PA5, following the orientation shown in figure 4.11. Solde the two 4 pins male bars in the places indicated for JP9 and JP10. Solde the 16 pins socket according to the board lines, and finally solde the 19 pins male bars to the board. Insert the 74HC4051 integrated circuit in the socket (half-moon with half-moon).

Figure 4.11 shows an analog interface board assembled. All the bars pins of the PortA (analog Port) are accessible. The 1 to 5 pins from PortA are directly accessible, and the bolt of the same port multiplexed with 8 inputs.

Front side

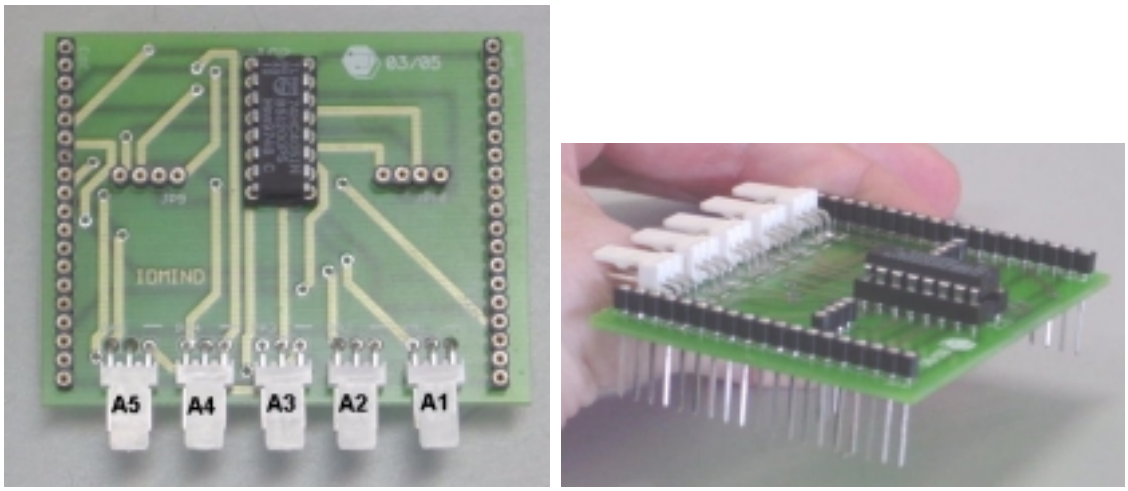


Figure 4.11 – Analog Interface Board

4.3.4 Construction of the Sensors Base Board ID-BASEIR002

Separate the ID-BASEIR002 board (fig. 4.12). For such, cut the small connections that join it to the set of boards and, if necessary, use a fine sandpaper. Identify the components used in this board

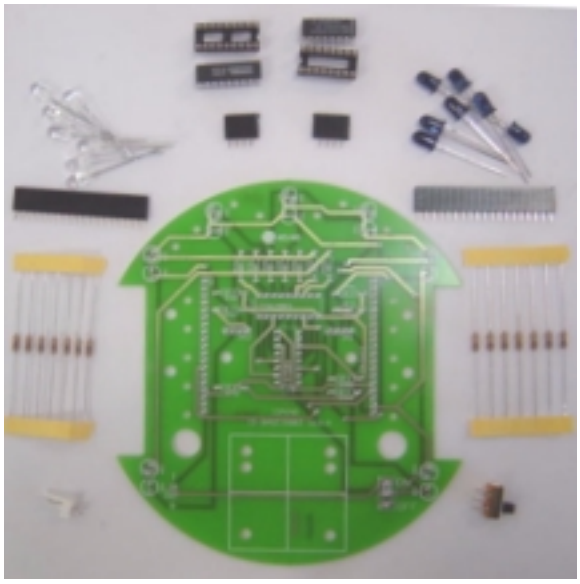


Figure 4.12 – Components used in the construction of the ID-BASEIR002 board

Begin the construction of this module by soldering the 7 resistors of 100Ω (RIR0 to RIR6), and the 7 resistors of 10KΩ (RTR0 to RTR6). Solde the two 16 and 18 pins sockets according to the board lines. Solde the 2 bars of 4 pins female in the places indicated for JP11 and JP10. Solde the 19 pins female bars to the board. Solde the on-off interruptor.

Identify the infra-red emitters and receivers, and its polarization (section 4.2.4). Identify where they must be connected. The emitters (lighter) must be soldered in the places indicated for IR0 to IR6. The receivers (darker) must be soldered in the places indicated for TR0 to TR6. These sensors must be connected bewaring of its purpose.

For example, if the purpose is to detect obstacles around the robot, the sensors must be placed on the top of the board. If the purpose is to detect a line painted on the floor, they must be placed beneath the board.

Figure 4.14 shows an example of assembly having the three pairs emitter/receiver of the front (IR2 IR4 and TR2 to TR4) placed beneath the board in order to follow a line painted on the floor.The remaining sensors were placed on the board top for obstacles detection.

Notice (figure 4.14) that the sensors placed up side down must be soldered bewaring of the distance from the board to the sensor extremity (this distance must be between 2,5 and 3 cm). Place the integrated in the respective sockets with the correct orientation (half-moon with half-moon).

Figure 4.13 – Battery support assembly

Solde the white male terminal in JP8. Place the battery support as shown in figure 4.13. For such, solde the small metallic terminals with the wires of the support, and insert them in the respective white support (the one with 2 holes). This support will be connected to the power terminals “+”and “-“of the ID-

BASEIR002 board. Beware that, when making the connection, the red wire corresponds to the positive terminal.

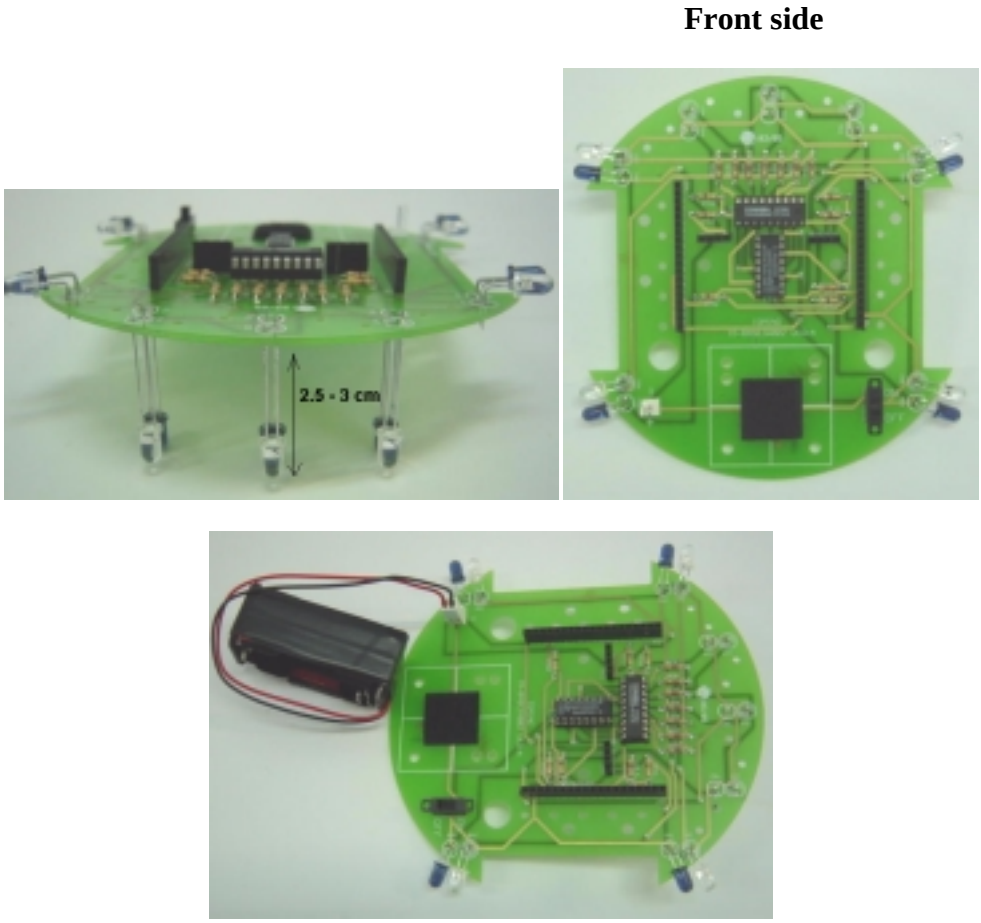


Figure 4.14 – Front side of the ID-BASEIR002 board

If you want to connect, on the front side of the robot, the micro-switches included in the kit, you must start soldering one of the sides of the three wire cables with the micro-switches, and on the other side the small metallic terminals supplied in the kit.

Insert the terminals in the white support (the one with 3 holes). Beware that the common terminal, represented as "COM" in the micro-switch, has to be inserted in the left terminal of the white support, when the two small plastic pins of the support faces the top (figure 4.15).

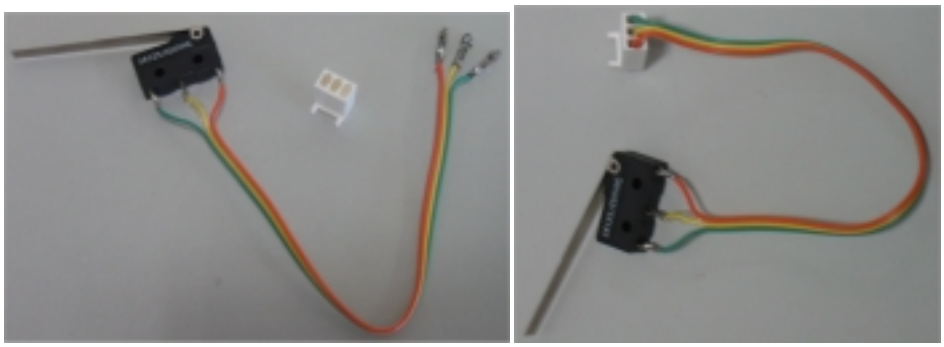


Figure 4.15 – Micro-switch assembly

Insert the 2 Micro-switches in the ID-BaseIR002 board. Insert the four screws in the holes found at the board front, close by IR4 and IR2 (figure 4.16).

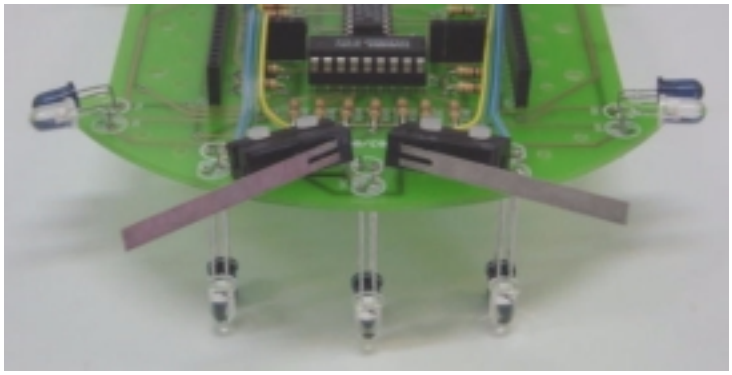


Figure 4.16 – Micro-switches localization

4.3.5 Robot different boards interconnection

This section shows how the robot different boards and components must be interconnected. Place the motors and the free wheel as shown in figure 4.17.

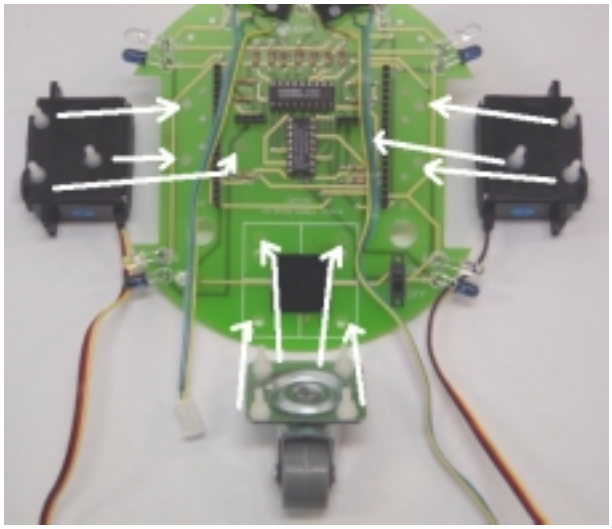


Figure 4.17 – Motors and free wheel

The two motors must be placed beneath ID-BASEIR002 board. For each motor there are five holes, but only the rear three must be used (figure 4.17). Place the free wheel as shown in figure 4.17 (use four holes). Insert the servo motors wires into the holes located in the board central part (figure 4.18).

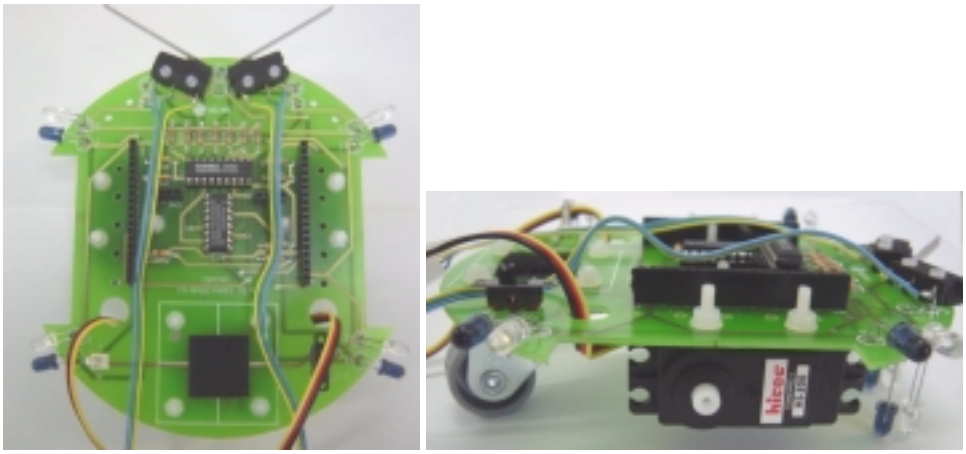


Figure 4.18 – Motors and free wheel assembly

Place the wheels on the servo motors and fasten the screw in its centre (figure 4.19).

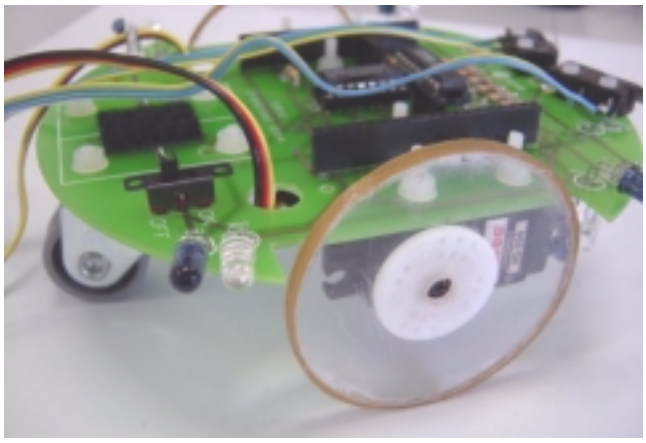


Figure 4.19 – Motors and wheels assembly

Insert the ID-AS002 board into the ID-BASEIR002 board. Verify if all the pins of the male bars can be inserted in the respective female bars (19 and 3 pins bars). Press both boards until they are completely inserted (figure 4.20).

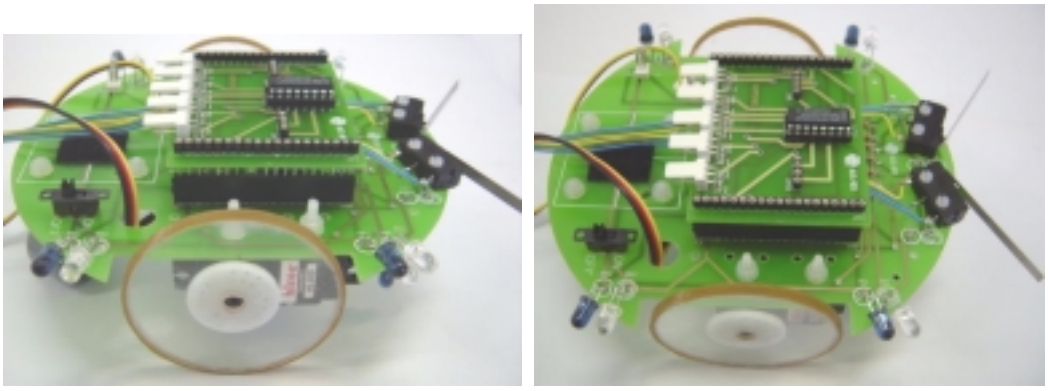


Figure 4.20 – Inserting of ID-BASEIR002 and ID-AS002 boards.

Connect the micro-switches terminal plugs: the left micro-switch in the terminal PB4, and the right in terminal PB3 of the ID-DS002 board. Connect the two motors terminal plugs in the PWM output: right motor in PWM1 output and left motor in PWM2 output. Beware of the black wire (mass) which must be turned to left side, when the robot shows its back to the user (figure 4.21).

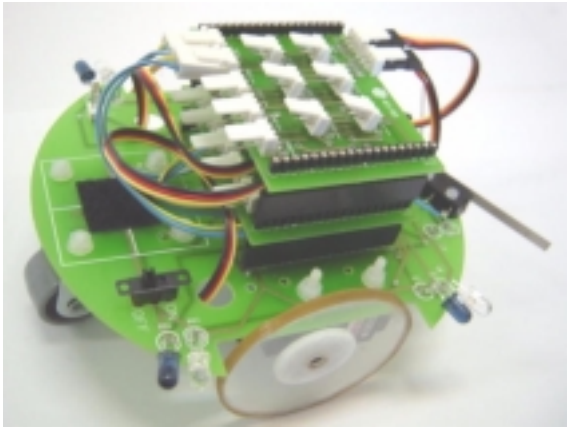


Figure 4.21 – ID-DS002 board encasement

Finally encase the ID-PIC002 board in the ID-DS002 board. Press both boards until they are completely inserted. At the end the robot must look like the one shown in figure 4.22.

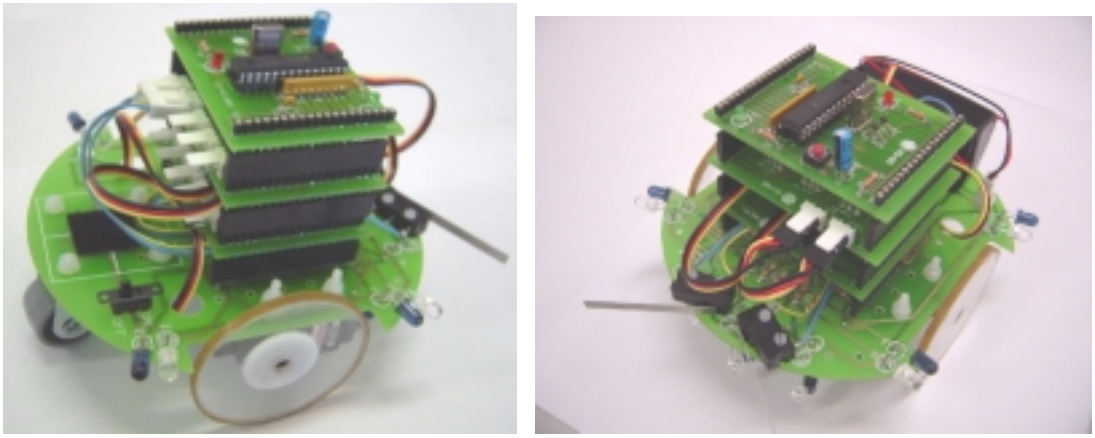


Figure 4.22 – Robot assembly

As described in section 3.2, the infra-red sensors occupy only one of the five analog inputs of the microcontroller. Four analog inputs are still available, and the user can used them to add another analog sensors.

The connection of new sensors of analog impute is very easy, being enough to bind the output bolt of the new sensors to the input of available analog signal at the robot, more concretely in the analog interface ID-AS002 board.

The digital interface ID-DS002 board makes available 11 digital input/output that can be used to increase the number of sensors/actuators. The 2 micro-switches occupy two of those entrances.

4.3.6 Program Board ID-PROG002 Assembly

Separate the ID-PROG002 board (fig. 4.23). For such, cut the small connections that join it to the set of boards and, if necessary, use a fine sandpaper. Identify the components used in this board

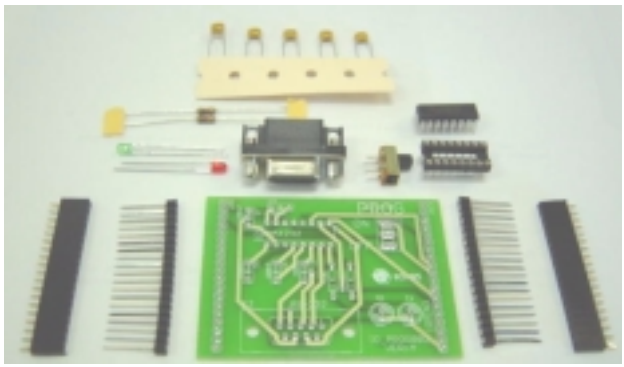


Figure 4.23 – Components used in the construction of the ID-PROG002 board

This board allows programming the robot through the serial port of a computer.

Solde the two type 1KΩ resistors. Solde the 16 pins socket beware of the board lines. Both leds must be soldered in TX (green) and RX (red), bewaring of its polarity (section 4.2.4).

Solde the five 100 nF capacitors (C5 to C9). Solde the two 19 pins male bars. Solde the on-off interruptor. Solde the serial port connector in the indicated place as RS-232. Finally, encase the two 19 pins female bars in the male bars, and place the 232 integrated circuit in the respective socket (half-moon with half-moon). By the end of the assembly, the board must looks like as shown in figure 4.24.

Front side

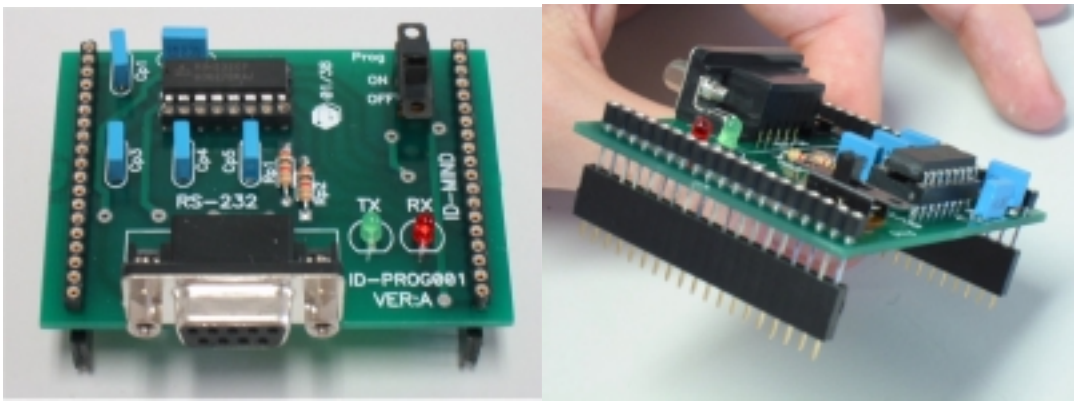


Figure 4.24 – Lateral view of the ID-PROG002 board

5 Microcontroller

The used microcontrollers are a PIC 16F876 family from Microchip. These *chips* present advantages over the majority of the microcontrollers, since they contain several functional blocks allowing to perform a huge number of tasks without the necessity of additional use of electronics.

In this chip it is possible to find A/D converters, temporizers, serial and parallel communication, PWM outputs, interruptions, etc (figure 7.1).

It contains an EEPROM memory of 8 K words of 14 bits, and a memory of data RAM of 368 words of 8 bits. The microcontroller can be programmed by the user, through the use of a low-level language, assembly type, of only 35 basic instructions, using, for such, the Mplab builder supplied for the Microchip. (http:\\www.microchip.com) in the page:

http://www.microchip.com/10/tools/picmicro/devenv/mplabi/software/v540/index.htm

The programming of this chip is made by using both the ID-PROG002 board, and the **IdMind**' **Loader1** program, that loads the builded file (*.hex) by **Mplab**.

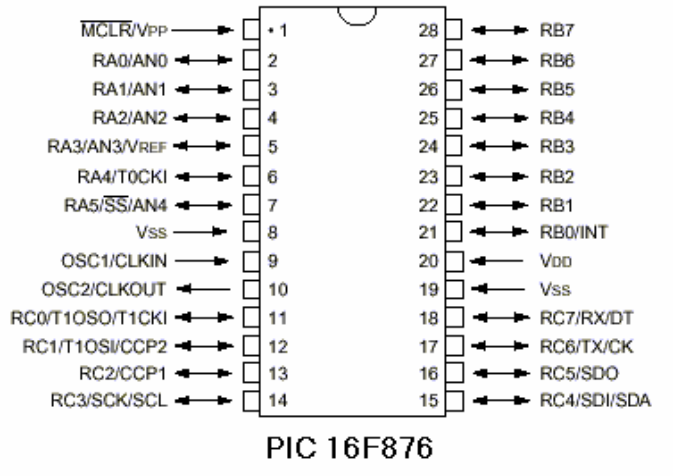


Figure 5.1 – PIC 16F876 Microcontroller

The inputs and the outputs are organized as shown:

PortA → 6 digital inputs/outputs or 5 analog inputs

PortB → 8 digital inputs/outputs

PortC → 8 digital inputs/outputs, being able to have 2 PWM outputs

This microcontroller has internal records, which allow to verify the situation of any running operation. For example, it is possible to verify if a subtraction outcome is positive or negative, to verify the elapsed time since a temporizer was turned on, to verify if an analog reading was already made, or even allowing the alteration of an output port to an input port.

These records are stored inside four memory banks (*Bank* 0, 1, 2 e 3). The table in figure 5.2 shows the records distribution by the several banks. Later it will be shown the most important records for the initiation of these microcontrollers.

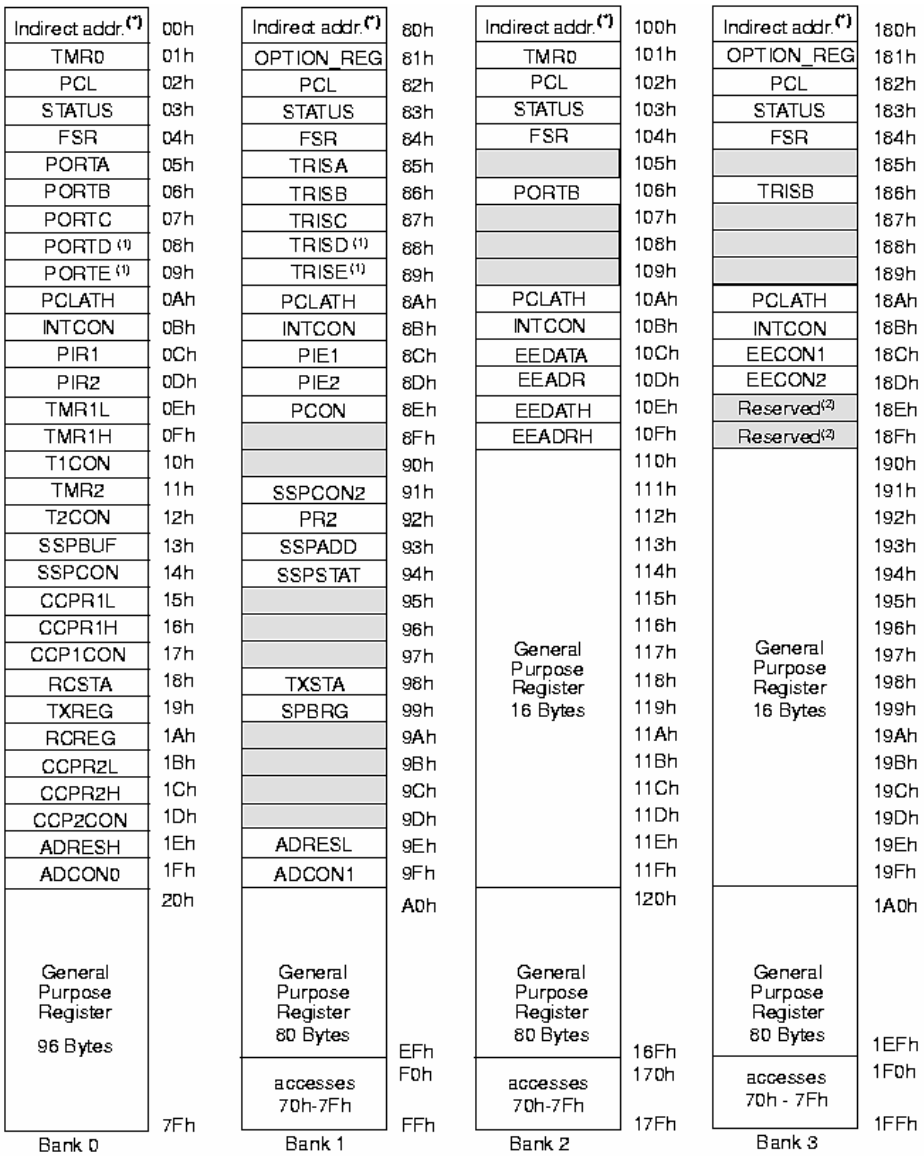


Figure 5.2 – Records distribution

One of the most important records is the **STATUS** record. It is this record that allows to define the Bank where we are located, and verify if the outcome of an arithmetical operation or logical is equal to 0 or negative.

STATUS Record

IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C

bit7 bit0

RP1:RP0: Bank bits of selection

11 = *Bank* 3

10 = *Bank* 2

01 = *Bank* 1

00 = *Bank* 0

Z: *Bit* Zero

Se **Z**=1 then, the last arithmetic or logical operation was equal to 0.

Se **Z**=0 then, the last arithmetic or logical operation was different of 0.

C: *Carry* (more than 8 bits utilization)

C=1 in the case of *Carry*

C=0 otherwise

For a better programming understanding, and for the utilization of record manipulation introductory examples, it is shown in the next table, the microcontroller instructions. In the right column it is indicated the record STATUS alterations.

Operation	Description	STATUS
Operations with records words (d =0 result in w, d=1 result in f)		
ADDWF f,d	W+f	C, Z
ANDWF f,d	W AND f	Z
CLRF f	f = 0	Z
CLRW	W = 0	Z
COMF f,d	Complement of f	Z
DECF f,d	f - 1	Z
DECFSZ f,d	f - 1 skip next operation if f -1=0	
INCF f,d	f + 1	Z
INCFSZ f,d	f + 1 skip next operation if f +1=0	
IORWF f,d	w OR f	Z
MOVF f,d	w = f or f = f	Z
MOVWF f	f = w	
NOP	It does nothing	
RLF f,d	rolls 1 bit left by Carry	C
RRF f,d	rolls 1 bit right by Carry	C
SUBWF f,d	f - w	C, Z
SWAPF f,d	Switch the first 4 bits with the 4 last	
XORWF f,d	f XOR w	Z
Operations with records bits (b=0 to 7 number of the bit to be altered or tested)		
BCF f,b	bit b from record f equal to 0	
BSF f,b	bit b from record f equal to1	
BTFSC f,b	Test bit b from f if equal to 0 skip next Operation	
BTFSS f,b	Test bit b from f if equal to 1 skip next Operation	

Operations with literals (k number from 0 to 255 in Math operations and from 0 to 2047 in subroutines)		
ADDLW k	w=w+k	C, Z
ANDLW k	w=w AND k	Z
CALL k	The Subroutine k is called	
CLRWDT	watchdogtimer in zero	
GOTO k	Go to Subroutine k	
IORLW	w = w OR k	Z
MOVLW k	w = k	
RETFIE	Return of an interruption	
RETLW k	Return with a literal in w	
RETURN	Return of a Subroutine	
SLEEP	Enters in Standby	
SUBLW k	w = k - w	C, Z
XORLW k	w = w XOR k	Z

As shown, the arithmetic and logical operations can provoke alterations in the **STATUS** record.

Having the knowledge of both this record and the instructions it is possible to start making some program blocks. For example, to configure the several Ports (A, B and C) as inputs, outputs or mixed.

5.1 Inputs/Outputs Configuration

In order to configure each microcontroller pins as an input or an output, it is necessary to configure, in the appropriated internal records, the way how these pins must be used. The configuration of each **Port** is made through the configuration of the correspondent **Tris**, which is in the Bank 1.

The process is the following:

1. Go to Bank 1
2. Configure the **Port**s writing in the **Tris** (**TrisA**, **TrisB** e **TrisC**)
3. In order to configure them as outputs, the **Tris** bits must have the value 0
4. In order to configure them as inputs, the **Tris** bits must have the value 1
5. Go back to Bank 0

Configuration example:

```
IN_OUT:                 ; Outputs and inputs configuration
        bsf STATUS,RP0  ; go to Bank 1
        bcf STATUS,RP1
        movlw .255      ; configure PortA as inputs
```

```
        movwf TRISA
        movlw .0
        movwf TRISC             ; configure PortC as outputs
        bsf TRISC,7             ; configure bit 7 of the PortC as output
        movlw .255
        movwf TRISB             ; Port B initialization as inputs
        bcf TRISB,7             ; configure bit 7 of the Port B as output
        bcf TRISB,5             ; configure bit 5 of the Port B as output
        bcf TRISB,3             ; configure bit 3 of the Port B as output
        bcf TRISB,1             ; configure bit 1 of the Port B as output
        bcf STATUS,RP0          ; bank 0
return
```

Another function of the **PIC,** which is going to be used is the acquisition and the conversion of an analog signal in a digital signal (word of 8 bits). The next section explains how the AD converter (Analog/Digital), existent in the microcontroller, functions internally.

5.2 Analog Inputs

The conversion module A/D of the PIC 16F876 is constituted by 5 inputs, all of them multiplexed, as shown in figure 5.3.

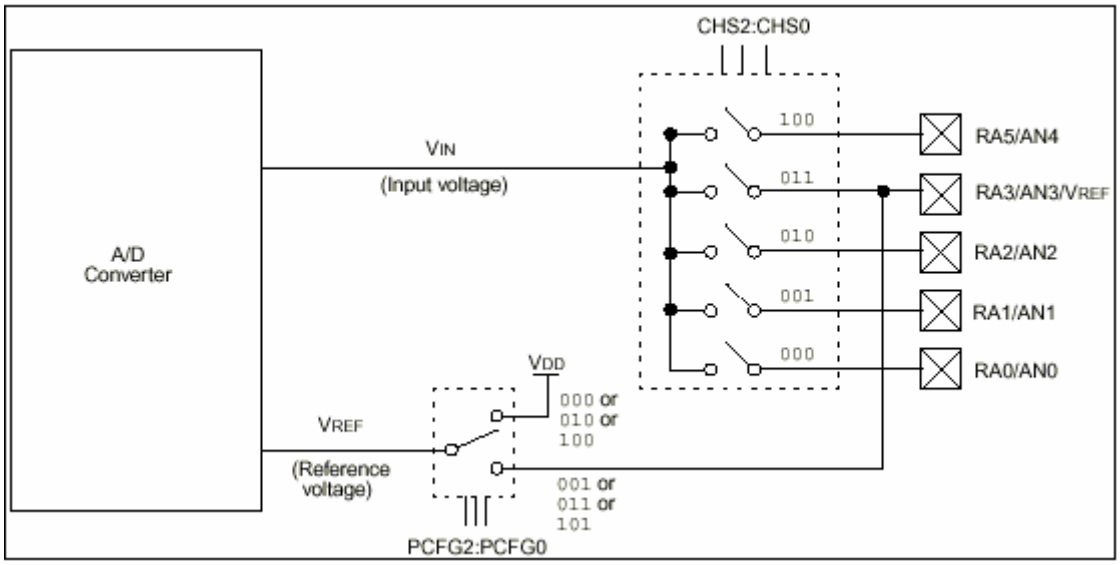


Figure 5.3 – Internal Diagram of the PIC converter A/D

This module allows the conversion of an analog signal in a corresponding digital signal of 8 bits. There is the possibility to use a voltage of external reference, correspondent the maximum value (255) of

conversion to this voltage of reference. If any voltage of reference is used, the power voltage is used as the value of reference.

In order to program this module it is necessary to configure three new records, which are now introduced.

ADRES – Conversion A/D outcome

ADCON0 – Control 0 Record

(It allows the choice of the conversion time, the channel that is going to be converted,

execute the conversion and verify if the conversion was already made)

ADCON1 – Control 1 Record

(It allows the Port A inputs configuration as Analog or Digital)

5.2.1 **ADCON0**

bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	--	ADON

ADCS1:ADCS0: Selection bits of conversion frequency

00=**Fosc**/2

01=**Fosc**/8

10=**Fosc**/32

11=**FRC** (Frequency of the internal RC of the oscillator)

Fosc is the crystal frequency used in the **PIC** assembly, in this case 4 MHz. It is necessary the configuration of this frequency in order to secure that this conversion is correctly made. The table 5.1 shows the temporal relation to secure that proper conversion.

AD Clock Source (TAD)		Device Frequency			
Operation	**ADCS1:ADCS0**	**20 MHz**	**5 MHz**	**1.25 MHz**	**333.33 kHz**
2TOSC	00	100 ns(2)	400 ns(2)	1.6 µs	6 µs
8TOSC	01	400 ns(2)	1.6 µs	6.4 µs	24 µs(3)
32TOSC	10	1.6 µs	6.4 µs	25.6 µs(3)	96 µs(3)
RC(5)	11	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1)

Table 5.1 – Choice of the Conversion temporizer

It is necessary to choose the analog bolt whose signal the user wants to convert. Remember that, although there are 5 analog inputs, these are multiplexed, and it is possible to convert only one signal at once.

CHS2:CHS0: Selection *Bits* of the channel to be converted
000 = channel 0, (RA0/AN0)
001 = channel 1, (RA1/AN1)

010 = channel 2, (RA2/AN2)
011 = channel 3, (RA3/AN3)
100 = channel 4, (RA5/AN5)

The next bit (bit 2) says to the processor to make the conversion, and, as an answer, it is possible to test the same bit to know if the conversion has already been made.

GO/DONE: Bit of the conversion station

If ADON=1
and
If **GO/DONE**=1 - Conversion A/D being made.
If **GO/DONE=** 0 - Conversion A/D stopped. (This bit is automatically 0 if the conversion is being made)

The next bit allows this module to switch on/off.

ADON: Activates the A/D conversion module
ADON=1 – The converter A/D is ready to operate
ADON=0 – The converter A/D is turned off and saves energy.

The next record, as it was pointed out previously, indicates to the processor which microcontroller pins are going to be used for both analog and digital signals, and still allows to configure the existence or not of a reference voltage

ADCON1

bit 7					**bit 2**	**bit 1**	**bit 0**
--	--	--	--	--	PCFG2	PCFG1	PCFG0

PCFG2:PCFG0	**RA0**	**RA1**	**RA2**	**RA3**	**RA5**	**Vref**
000	A	A	A	A	A	VDD
001	A	A	A	A	Vref	RA3
010	A	A	A	A	A	VDD
011	A	A	A	A	Vref	RA3
100	A	A	D	D	A	VDD
101	A	A	D	D	Vref	RA3
11x	D	D	D	D	D	-

Example of an A/D conversion, in channel 0:

Initialization:

```
bsf  STATUS,RP0        ; chooses the bank 0 od the STATUS record
bcf STATUS,RP1         ; chooses the bank 1 od the STATUS record
clrf ADCON1            ; configures the A/D Inputs
bcf  STATUS,RP0        ; goes back to bank 0
movlw b'01000001'      ; Fosc/8, A/Dis active, channel 0 selected
movwf ADCON0
bsf ADCON,2            ; initiates the conversion
```

Converte:

```
btfsc ADCON0,2          ; verifies if the conversion is over
goto Converte           ; not yet,  verify again, until is finished
movf ADRESH,0           ; the value is in the ADRESH record
```

5.3 PWM (Pulse Width Modulation) Outputs

The PWM signal is a rectangular wave of constant period, but it has the possibility to have one variable Duty-cycle. The Duty-cycle is considered as being the percentage of time in a period of time in which the wave value is maximum. Figure 5.4 shows an interpretation of the PWM wave.

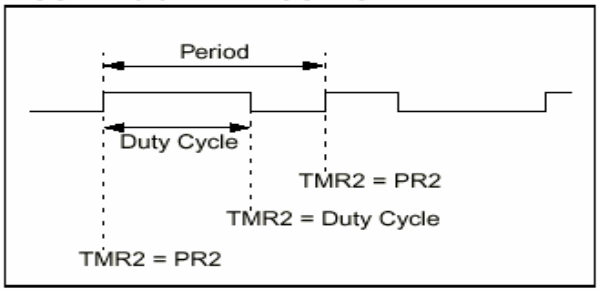


Figure 5.4 – PWM wave

The chosen PIC has the capacity to have already two modules of PWM outputs. These modules are very important, since they can be used to actuate the robot motors, allowing to vary the speed of each wheel. These two PWM modules can have up to 10 bits resolution. To program these modules it is necessary to choose a **PWM Period**, which should not be modified, and later choose the Duty-Cycle, which can be altered as desired.

5.4 PWM Period

To start, the user must choose a value for the period of the **PWM** wave. Later, it must be solved the following equation in order to **PR2** (which is an 8 bits record).
Period of PWM = [(**PR2**) +1)] x 4 x **Tosc** x (**TMR2 prescale value**)
The **TMR2** (**prescale value**) is one of the three possible values 1, 4 or 16 and the **Tosc** is the inverse of the **Fosc** frequency.

5.4.1 PWM *Duty-Cycle*

While making the program the user can have the necessity to modify the Duty-Cycle wave.
For this effect, it must be used the following equation:

Duty Cycle do PWM = (**CCPR1L:CCP1CON<5:4>)** x **Tosc** x (**TMR2 prescale value**)

5.4.2 Maximum number of *Bits*

How greater it is the PWM frequency, the closer is of the frequency of the crystal oscillation, being limited to the use of only some bits of conversion, instead of 10 bits. To calculate how many bits can be used in the conversion, it is enough to execute the following operation

$$\text{Maximum number of bits} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

In order to get a bigger resolution it is necessary to diminish the frequency. To get a greater frequency it is necessary to diminish the resolution.

The next table shows some frequencies in which are known the **PR2** values, the **TMR2 prescale value**, and its resolution.

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12 kHz	156.3 kHz	208.3 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Maximum Resolution (bits)	10	10	10	8	7	5.5

Table 5.2 – PWM periods and corresponding configurations

In this module it is introduced some new records that will be studied. Previously it has been introduced the existence of three temporizers. In this case it is used the Temporizer 2 (**Timer2**), to estimate the wave period value, for comparison with the value defined by **PR2**.

PR2: Programming of the PWM Period

T2CON: Programming of the**TMR2 prescale value**

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
--	--	--	--	--	TMR2ON	T2CKPS1	T2CKPS0

TMR2ON: Activates the *Timer 2* temporizer

TMR2ON =1 - *Timer 2* active

TMR2ON =0 - *Timer 2* inactive

T2CKPS1: T2CKPS0: selection *Bits* of the TMR2 prescale value

T2CKPS1: T2CKPS0 = 00 - *prescale value* equal to 1

T2CKPS1: T2CKPS0 = 01 - *prescale value* equal to 4

T2CKPS1: T2CKPS0 = 1x - *prescale value* equal to 16

CCPR1L e CCP1CON<5:4>: Programming of *Duty-Cycle*

O **CCPR1L** record corresponds to the more significant 8 *bits*, and the **CCP1CON<5:4>** to the less significant 2 *bits*.

5.4.3 Steps for PWM initialization

1. Initialize the **PWM** period writing in the **PR2** records;
2. Initialize the **PWM** *Duty-Cycle* writing in the **CCPR1L** e **CCP1CON<5:4>** records;
3. Transform the **CCP1** and **CCP2** *bits* in outputs, writing 0 (zero) in the *bits* 1 and 2 of **TrisC;**
4. Establish the **TMR2 prescale value** and activate the *Timer 2* writing in **T2CON;**
5. Configure the **CCP1CON** and **CCP2CON** modules for **PWM** operations, writing the binary value b'00001111' in both records.

Example of the PWM programming:

```
bsf STATUS,RP0          ; go to bank 0 to program the PR2
movlw .255              ; program the PR2 record (program the Period)
movwf PR2
bcf STATUS,RP0          ; go back to bank 0
movlw .12               ; program the 8 bits of the CCP1CON e CCP2CON records
movwf CCP1CON           ; with the value 12 (Program the Duty Cycle)
movwf CCP2CON
movlw .5                ;Activate Timer 2 and use the prescale value to 1
movwf T2CON
movf Andar_Esquerda,0
movwf CCPR2L
movf Andar_Direita,0
movwf CCPR1L
```

5.5 Table Utilization

In several situations, to diminish the computational weight, the use of tables with predefined values becomes very important. In the case of our robot, the use of these tables becomes very important to define some behaviours that the robot must present in determined situations.

For example, if the robot knows that it is perfectly lined up with the line painted on the floor, then it can run faster. Another example is the following three different situations: the robot has the line on its side, or the line is in the middle or the line is on the other side. In this in case it is possible to define these situations in a table and in another two tables corresponding to the **PWM** values that must be sent for each one of the motors.

Example:

Pista:

```
        addwf PCL, 1            ; Table that illustrates the three possible robot situations
        retlw b"11100000"
        retlw b"00111000"
        retlw b"00000111"
```

MotorDireito:

```
        addwf PCL, 1            ; Table that illustrates the PWM values for the motor
        retlw .10               ; from the right to one of the three situations
        retlw .200
        retlw .255
```

MotorEsquerdo:

```
        addwf PCL, 1            ; Table that illustrates the PWM values for the motor
        retlw .255              ; from the left to one of the three situations
        retlw .200
        retlw .10
```

To call each one of these tables, it is only necessary to make one **call** - introduce a value in the (w) record. If w=0 corresponds to the first element of the table, then w=1 corresponds to the second element of the table, and etc.

```
Movlw .2                    ; w=2
call MotorDireito           ; Calls the MotorDireito routine and returns the value
------------------          ; w = 255
```

5.6 Interruptions

In several cases there is the necessity to have some events that must be treated when they occur, for such the user can create **interruptions** that are treated, in first place, outside the normal cycle of the program. These interruptions can be generated by signals in some of the microcontroller pins, or they can be created by condition alterations of some records. The following program shows how it is possible to activate an interruption whenever the **Timer2** is equal to the value of **PR2** record.

; initialization of the interruptions records

```
        movlw  b'11000000'
        movwf  INTCON
        bsf    STATUS,RP0
        bcf    STATUS,RP1
        movlw  b'00000010'
        movwf  PIE1
        bcf    STATUS,RP0
        bcf    STATUS,RP1
```

5.6.1 INTCON Record

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

As shown in the block of the previous program, only the two more significant bits have the value 1. When the **GIE** bit assumes the value 1 it activates the interruptions, and the **PEIE** bit activates the interruptions of Timer2 type. The other bits not mentioned allow other kind of interruptions, signals of other timers and external signals.

5.6.2 PIE1 Record

bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE

In this case **TMR2IE** is active, and creates an interruption whenever *Timer2* is equal to **PR2** record.

```
; Interruptions Treatment

interruptions:

        movwf dabliu            ; Before, it must be saved all the information
        swapf STATUS,w          ; first, the information in w, and then in de record
        movwf statos            ; STATUS. The dabliu and statos records have to be
                                ; defined at the beginning of the program

        bcf STATUS,RP0
        btfsc PIR1, TMR2IF      ; Verify if the interruption referring to tmr2
        goto testa_tmr2         ; was the timer2 interruption, go to testa_tmr2
        bcf PIR1, 0             ; It was another interruption, go back to the main program
        goto fim_int

testa_tmr2:                     ; routine of treatment of the timer2 interruption
        bcf PIR1, 0             ; block the interruption
        goto PWM

PWM:
```

```
        movwf Anda_esquerda,0    ; make the desired treatment
        movwf CCPR2L             ; change the PWM1 and the PWM2 for example
        movwf Anda_direita,0
        movwf CCPR1L
        goto fim_int

fim_int:
        bcf PIR1, TMR2IF         ; clean the bit, it is already treated
        bsf STATUS, RP0
        bsf PIE1, TMR2IE         ; activate again the interruptions
        bcf STATUS, RP0
        movlw  b'11000000'       ; activate again the interruptions
        movwf INTCON

        swapf statos,w           ; go back to the old values of STATUS and w
        movwf STATUS
        swapf dabliu,w
retfie                           ; Go back to the initial program
```

PIR1 Record

bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF

This record allows verifying which of the interruptions activated by **PIE1** generated the interruption. In this in case the test is made to **TMR2IF** *bit*, since this corresponds to the active interruption of the *Timer2*. Always an interruption is detected, and after the treatment, this *bit* must assume the value 0.

5.7 Structure of a Program for the PIC

The programming of a microcontroller must, always possible, follow a determined order. This order is necessary, not only to allow its proper working, but also to allow its easy reading. In this section is shown an example for a program organization.

; Beginning of the program

; Configuration of the microcontroller and libraries to be used

```
LIST P=16F876
Include <P16F876.inc>
```

; Storing of memory in auxiliary variables

```
sensor1 EQU 0x20        ; The memory position 0x20 is reserved as
sensor2 EQU 0x21        ; sensor1, the position 0x21 as sensor2, etc
sensor3 EQU 0x22        ; The first value must always start with an address
sensor4 EQU 0x23        ; equal or superior to 0x20
sensor5 EQU 0x24
sensor6 EQU 0x25
sensor7 EQU 0x26
sensor8 EQU 0x27
contador EQU 0x28
AUX EQU 0x29
pista EQU 0x2A          ; 0x2A defines the value in six-decimal after the
                        ; 0x29
```

; Beginning of the Program

```
org  0x00               ; definition of the program position 0
goto Start              ; Jump to a desirable position of the program
                        ; beginning
org  0x04               ; Reserved zone for the interruptions
call interrupções       ; calls a subroutine of interruptions treatment
return                  ; Go back to the main program

org  0x10               ; Table zone
```

```
niveis:
        addwf PCL, 1
        retlw .50
        retlw .50
        retlw .50
        retlw .50
        retlw .50
        retlw .50
        retlw .50
        retlw .50
```

; Program main Cycle

```
        org 0x30        ; Defines the initial position chosen by the user
Start:
        call In_Out     ; Calls the routine that configures the PORT's

Ciclo:                  ; cycle that must be always repeating itself
        ----
        ----
        Any call
        ----
        goto Ciclo      ; repeats the cycle
```

; SubRoutines called during the main Cycle

```
qualquer:
        ------
        ------
        ------
return                  ; returns to the next line from where it was
```

```
                    ; called

; Routines of interruption treatment

interruptions:

        ------

        ------

        ------

retfie              ; Go back to the line that was being processed when

                    ; the interruption occurred

; Routine of input/output configuration


In_Out:

        --

        --

        --

        --

return


end                 ; there is no more code
```

6 Programming

In this chapter is given a special attention to the control system of the two servo motors of the infra-red board.

6.1 Program of the Infra-Red Board Control

Due to the characteristics of the Converter A/D of the used microcontroller, it is necessary a short time of loading of the converter capacitor, before initiating the conversion.

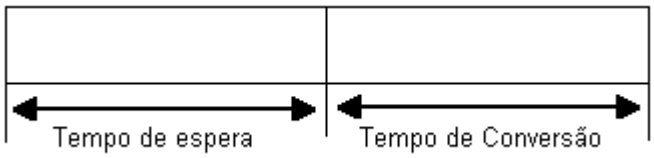


Figure 6.1 – Waiting time + Conversion Time

In this programming example it is used a waiting time equal to 3 ms for all *leds*, the conversion time is regulated by the microcontroller. The emitting time of the *leds* corresponds to the sum of the waiting time with the conversion time. Figure 6.2 shows the *leds* desired temporal configuration.

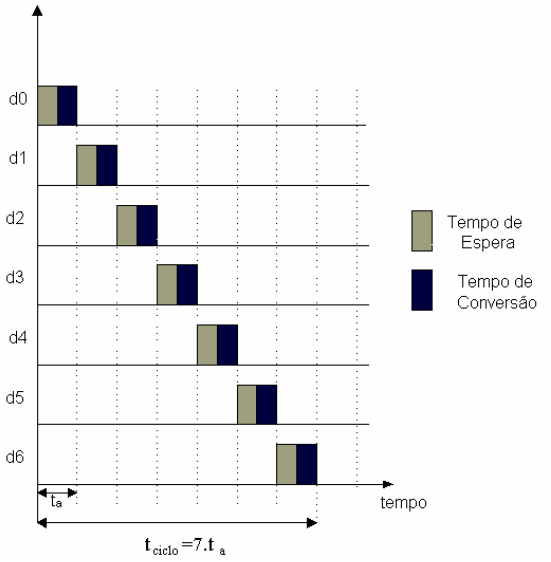


Figure 6.2 – Temporal Configuration

In order to actuate **S0**, **S1** and **S2** it was used the **PortC** pins 3, 4 e 5, and it was used the **PortC** bolt 0 to switch on the **Enable** of **Z**.

The analog values are sent to the ID-AS002 board, which makes the multiplexing of the 7 inputs infrared board, using the **PortC** 3, 4 and 5 bits, respectively. This multiplexing allows that only analog channel to be used, it is the **PortA** bolt 0.

	Bit 5	Bit 4	Bit 3
d0	**0**	**0**	**0**
d1	**0**	**0**	**1**
d2	**0**	**1**	**0**
d3	**0**	**1**	**1**
d4	**1**	**0**	**0**
d5	**1**	**0**	**1**
d6	**1**	**1**	**0**

Table 6.1 – Infrared emitters and receivers desired configuration

Program:

```
        LIST P=16F876

        include <P16F876.inc>

        Contadorleds EQU 0x24
        Contador EQU 0x2C
        Pista EQU 0x29
        ....

        org 0x00

        goto start

        org 0x04
        call interrupt
        return

org 0x10                    ; beginning of the table zone

limiar:                     ;Table of comparison tresholds with the analog values

        addwf PCL, 1
        retlw .70
        retlw .78
        retlw .55
        retlw .53
        retlw .30
        retlw .52
        retlw .48


org 0x30                    ; beginning of the program cycle

start:

        call In_Out         ; PORT'S initialization
```

```
        movlw b'00110001'       ; Beginnig of Timer1 to the waiting times, etc
        movwf T1CON

        call segue_linha:

goto start

segue_linha:

        bsf PORTC,0             ; bit Pc7 to one, turn on the infrareds
        bcf PORTC,3             ; bits Pc4,Pc5 e Pc6 at zero
        bcf PORTC,4
        bcf PORTC,5
        clrf Contadorleds       ; Count the number of already read leds
        clrf Pista              ; It contains the binary value of all leds



inicioconversao:

        movf TMR1L,0            ; Beginning of the waiting time
        btfss STATUS,2
        goto $-2

        movlw 0xFF              ; waiting time
        subwf TMR1L,0
        btfss STATUS,2
        goto $-3

        bsf ADCON0,2            ; Order to convert
        btfsc ADCON0,2          ; Conversion Time
        goto $-1

debug:

        movf Contadorleds,0     ; Number of read Leds
        call limiar             ; Comparison with a treshold (def.in the table)
        subwf ADRESH,0
        btfsc STATUS,0
        goto preto              ; In case it is under the treshold and black
        rlf Pista               ; Otherwise and White
        bsf Pista,0
        goto Contadores

preto:

        rlf Pista
        bcf Pista,0
        goto Contadores

Contadores:

        incf Contadorleds,1     ; Reads the next IV pair
        btfsc Contadorleds,3    ; Verifies if they are all read
        return

        movlw b'00001000'       ; Increases Pc4,Pc5,Pc6
        addwf PORTC,1

        goto inicioconversao

In_Out:
```

```
        bsf STATUS, RP0
        bcf STATUS, RP1
        bsf TRISA,0             ; PortA bolt 1 configuration as an input

        clrf PORTC              ; PortC all pins configuration as outputs
        bcf STATUS, RP0

        return

end
```

6.2 Control Program of the two Servo Motors

The used motors are modified servo motors. This means that the servos, which usually are used for angular control, are modified for velocity control. These are controlled by a **PWM** signal as shown in figure 6.3.

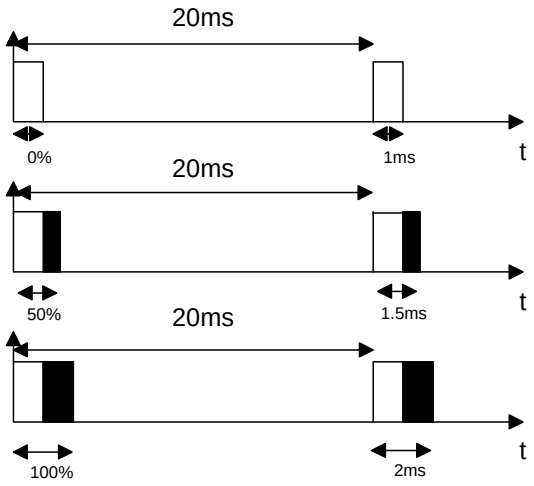


Figure 6.3 – PWM wave used in servo motors

As easily verified, this wave period is going to be 20 ms. This period has to be accomplished, since of it depends the proper working of these motors electronics. The **PWM** wave is decomposed in three different parts: a first zone in which the wave has the value 5 V through 1 ms; a second zone, which is called the ***duty-cycle***, since the servo velocity only depends of this zone, corresponding a *duty-cycle* of 100% to 1 ms; in the last zone the signal is taken to zero through 18 ms. Going back the *duty-cycle*, if it is 0% then the correspondent wheel will roll back with maximum velocity, if it is 50% the wheel will stop, and if it is 100% the wheel will roll ahead with maximum velocity.

The program that allows the signal realization to be sent to the two motors, divides the signal in 20 signals of 1 ms, being at 1 in the first millisecond. The second is modulated with the desired *duty-cycle* and it is turned to zero in the remaining 18 milliseconds. In order to divide the signal in 1 ms intervals, it was programmed the **PWM** of 1 ms. Always the 1 ms counting ends, it is generated an interruption, which, when treated in the interruption routine, chooses which is the next *duty-cycle* (100%, *duty-cycle* or 0%). The next program shows how it is possible to generate this kind of signal.

```
LIST P=16F876
include <P16F876.inc>

dabliu EQU 0x20              ; auxiliary variable that keeps the W
AUXSTATUS EQU 0x21           ; auxiliary variable that keeps the STATUS
ContadorPWM EQU 0x22         ; 20 cycles counter of PWM
Anda_direita EQU 0x23        ; variable that keeps the next value of PWM of the right motor
Anda_esquerda EQU 0x24       ; variable that keeps the next value of PWM of the left motor
org 0
goto 0x46

org 0x04
        call interrupt       ; Interruptions call
        return

org 0x46                     ; Beginning of the program configuration
        call In_Out          ; PORT's initialization
        movlw b'11000000'
        movwf INTCON         ; Activate the interruptions
        bcf STATUS,6
        bsf STATUS,5
        movlw b'00000010'    ; configure the interruptions
        movwf PIE1
        movlw .249           ; PWM period configuration
        movwf PR2
        bcf STATUS,5
        movlw b'00000101'    ; Prescale value
        movwf T2CON
        movlw b'00001111'    ; PWM mode configuration in the bolt 3 of PortC
        movwf CCP1CON
        movlw b'00001111'    ; PWM mode configuration in the bolt 2 of PortC
```

```
        movwf CCP2CON
        movlw b'00110001'
        movwf T1CON
        movlw .19               ; initialization of the number of 1 ms cycles, 19+1
        movwf ContadorPWM

start:                          ; demonstrative values to be sent to PWM's
        movlw .200
        movwf Anda_esquerda
        movlw .50
        movwf Anda_direita
        goto start

interrupt:
        movwf dabliu            ; All the relevant information must be saved before
        swapf STATUS,w          ; first the information in w, and then in the record of
        movwf AUXSTATUS         ; STATUS. The dabliu records must be
                                ; defined at the beginning of the program
        btfsc   PIR1,TMR2IF     ;Tests if it was the timer2 interrupton
        goto    testa_tmr2
        bcf PIR1, 0
        goto fim_int

testa_tmr2:
        bcf PIR1, 1             ; deactivates the interruptions
        movlw .19               ; verifies in which cycle it is
        subwf ContadorPWM,0
        btfsc STATUS,2
        goto PWM_100            ; in case it is in the first cycle the PWM stays at 100%
        movlw .18
        subwf ContadorPWM,0     ; in case it is in the second cycle it goes to program the PWM
```

```
        btfsc STATUS,2          ; otherwise the PWM is 0%
        goto PWM_PWM
        movf ContadorPWM,0
        btfsc STATUS,2
        goto Novacontagem
        clrf CCPR1L
        clrf CCPR2L
        decf ContadorPWM
        goto fim_int

PWM_100:                        ; in the first cycle it must put the 2 PWMs at 100%
        movlw .255
        movwf CCPR1L
        movwf CCPR2L
        decf ContadorPWM
        goto fim_int

PWM_PWM:                        ; in the second cycle it is going to use the PWM written in the variables
        movf Anda_esquerda,0    ; Left motor value
        movwf CCPR2L
        movf Anda_direita,0     ; and right motor value
        movwf CCPR1L
        decf ContadorPWM
        goto fim_int

Novacontagem:                   ; in case of the end of the 20 cycles, goes back to the first
        movlw .19
        movwf ContadorPWM
        goto fim_int

fim_int:                        ; activates the interruption again
```

```
        movlw b'11000000'
        movwf INTCON

        swapf AUXSTATUS,w       ; go back to the old STATUS and w values
        movwf STATUS
        swapf dabliu,w
retfie

In_Out:
        bsf STATUS, RP0
        bcf STATUS, RP1
        clrf TRISC              ; PORTC configuration as outputs
        bcf STATUS, RP0
        return
end
```

7 Experience Example

This chapter exemplifies a small test of the robot potentialities. The necessary steps to program the robot are going to be presented, using the ID-PROG002 programming board.

Included in the kit there is a CD containing a test program, not builded (.asm), which the user can utilize to verify the proper robot working. This example is used in a robot assembled according with the procedure described in the previous sections, and its final aspect must be like the one shown in the figure 4.22. After its programming, the robot will be able to follow a line painted (light/dark) on the floor (dark/light), and detect the collision with an object by using the *micro-switches.*

Make a copy of the CD contents to a computer directory. Execute the program “Loaderstp.exe” in order to install the application that will allow the downloading of the programs to the robot.

Install, in a PC, the program Mplab from Microchip. Use the following address to download this software:

http://www.microchip.com/10/tools/picmicro/devenv/mplabi/software/v540/index.htm

After the program installation execute the program Mplab. Open the project “Circ3.pjt” (Project - Open Project), figure 7.1. Build the project (Project – Build All).

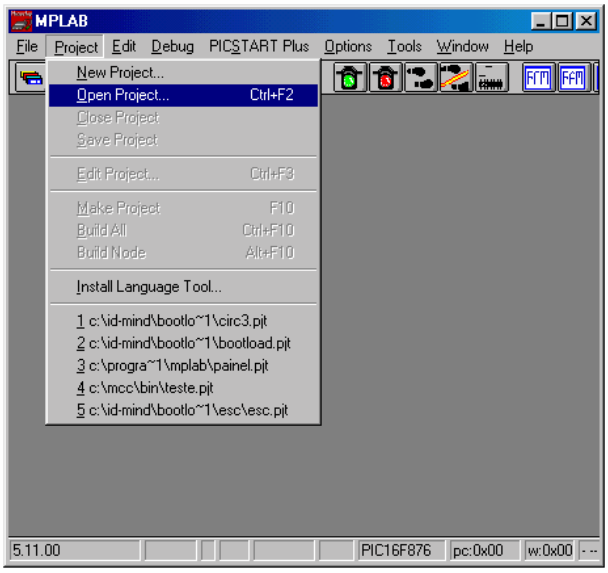


Figure 7.1 – Mplab from Microchip

After the building, it is generated, in the directory, the “CIRC3.HEX” file.

Insert the ID-PROG002 program board on the robot - on the top of the ID-PIC002 board. Connect the male terminal of the serial cable to the program board, and the female terminal to a serial input of the PC (figure 7.2).

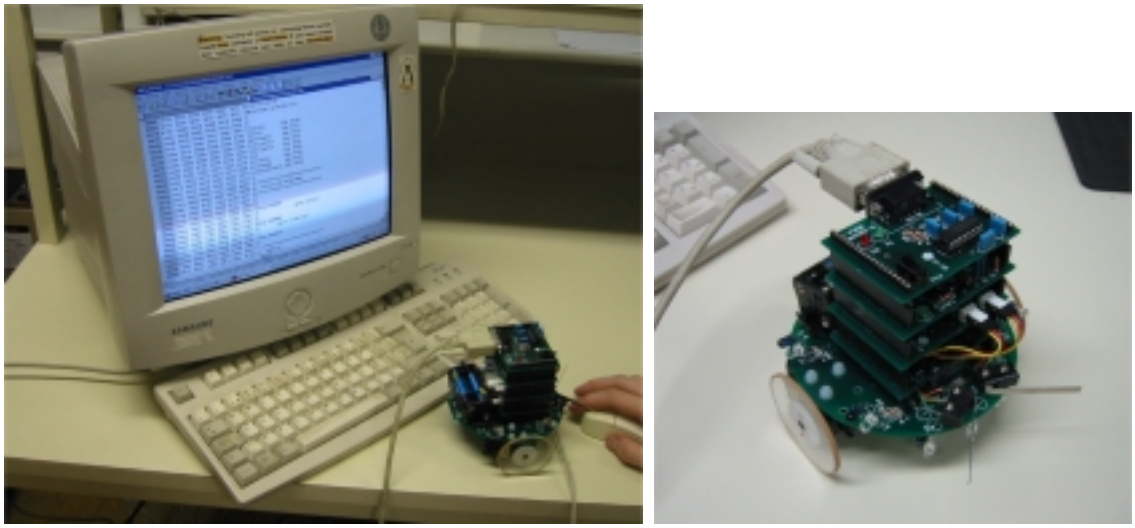


Figure 7.2 – Assembly to programming

Use now the program **Loader**, provided by IdMind, to download the builded program (.hex) to the robot microcontroller. For such, execute the program “Loader.exe” resulted from the “Loaderstp.exe” file installation. Configure, in the program, the serial port that is being in use (“com1” or “com2”).

Open the “CIRC3.HEX” file (File – Open), figure 7.3.

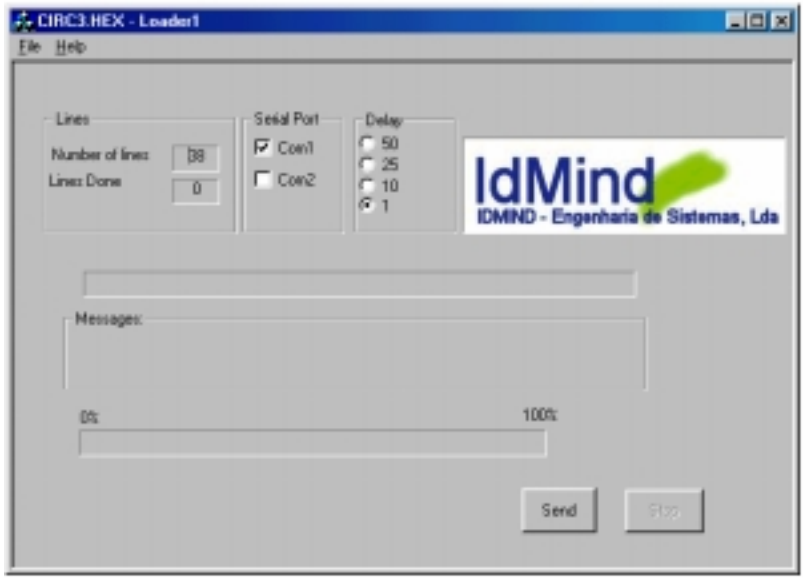


Figure 7.3 – Programming board Interface

Having the robot switched off (interruptor of the ID-BASEIR002 board “OFF”), switch the interruptor of the ID-PROG002 board to “ON”. Switch now the interruptor of the ID-BASEIR002 board to “ON”.

Press “Send” in the Loader Program and the program will start to be sent to the microcontroller (figure 7.4)

Figure 7.4 – Programming transaction

When the message “Success” appears, the robot can be disconnected (interruptor of the ID-BASEIR002 board turned “OFF”), and the interruptor of the program board can be disconnected to. Remove the serial cable from the program board.

In order to put the robot in movement, it is now enough to switch the interruptor of the ID-BASEIR002 board. The robot will be able to execute the programs with or without the program board inserted. In order to execute them, with the board inserted, its interruptor must be switched “OFF”. Pay attention that the execution of the programs with the board inserted will lead to a small increase of energy spent.

7.1 Error Messages

When the program is sent to the robot, it is possible to happen some errors due to inaccurated procedures.

Next, it is presented some usual errors, and related causes.

Error: “ERROR: Serial Port Selected not Enable!!! “

Cause: The chosen serial port has already a connected peripheral (mouse).

Solution: Choose the other “com” in the program

Error: “ERROR: The Pic does not respond ”

Cause: The robot is not switch “ON”

The control board interruptor is not switch “ON”

The serial port associated is not the one defined

Solution: Having the robot switched “OFF” (ID-BASEIR002 board switched “OFF”), switch “ON” the interruptor of the ID-PROG002 board. Switch now the interruptor of the ID-BASEIR002 board to “ON”. Verify the serial port or choose the other “com” in the program.

Error: “ERROR: The Pic can’t be programmed”

Cause: The computer data is not correct

The PIC can not process the computer data.

Solution: Build the data again

Increase the program “Delay”

Error: “ERROR: Stop by USER!!!”

Cause: The user stopped the download using the “Stop” button

Solution: Send again the “Send” program

Appendix A - Circuit Schematic

A1) Controller Block (ID-PIC002, ID-DS002 e ID-AS002)

Figure A.1 shows the electric scheme used in the controller set.

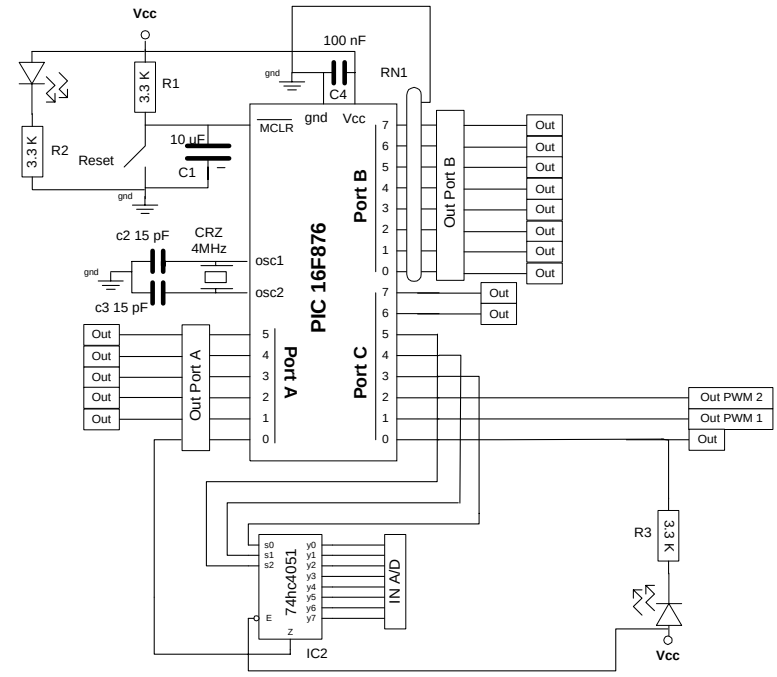


Figure A.1 – Controller block connection scheme

A2) InfraRed Block

The electric diagram of the infrared board (ID-BASEIR002) is shown in figure A.2.

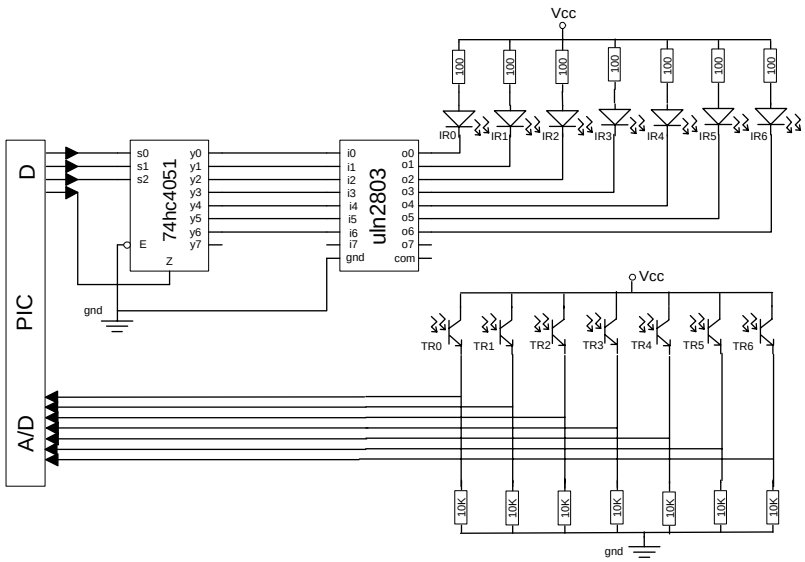


Figure A.2 – Infrared board' electric diagram

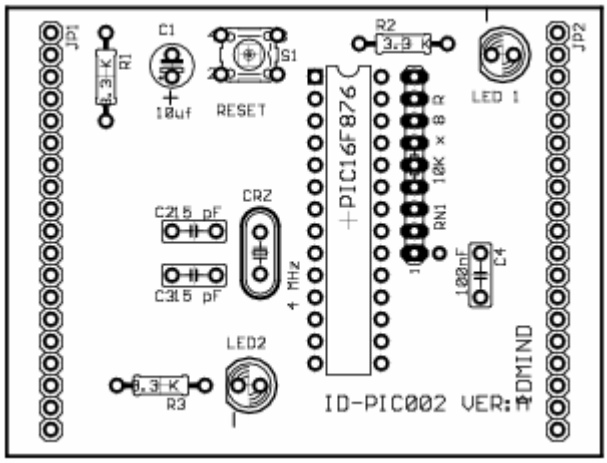
In this circuit are used four digital outputs of the PIC' PortC. The bits 3, 4 and 5 of this port are connected, respectively, to "S0", "S1", and "S2", and they define which of the leds is going to emit. The fourth output, bit 0 of the PortC, is connected to the output "z" allowing to connect or to disconnect the emitters.

The seven analog outputs coming from the infrared receivers are connected to the microcontroller analog inputs. This connection is made in the interface analog board that allows the multiplexing of these seven inputs and to read each one of them using only one of the analog inputs of the microcontroller.

Appendix B– Components and Boards

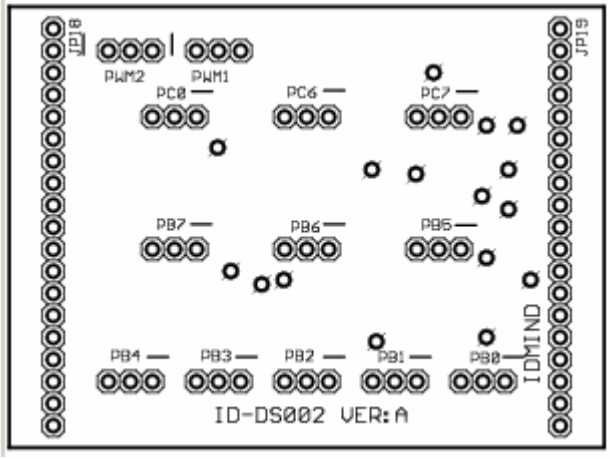
B1) Control Board - ID-PIC002

Description	Quantity
PIC 16F876	1
Socket 0,3" 28 way	1
100 nF Capacitor	1
3,3 K Resistor	3
8*1K Resistor (Pack of Resistors)	1
Green Led 3 mm	1
Red Led 3 mm	1
15 pF Ceramic capacitor	2
10 µF/50V Electrolytic capacitor	1
Switch (Reset)	1
Cr. 4 MHz	1
19 pins female bars	2
19 pins male bars	2



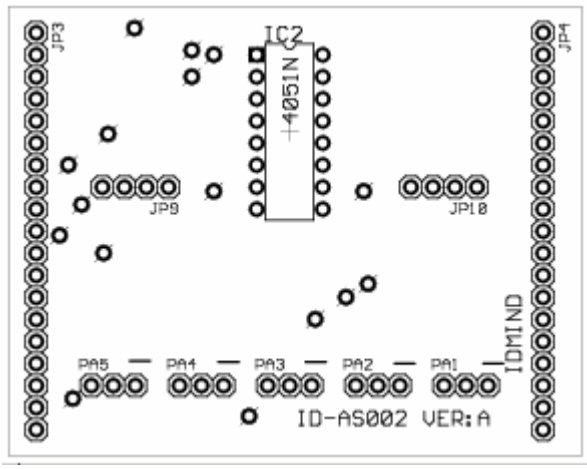
B2) Digital Interface Board ID-DS002

Description	Quantity
19 pins female bars	2
19 pins male bars	2
Connection pins at 90º 3-pins	13



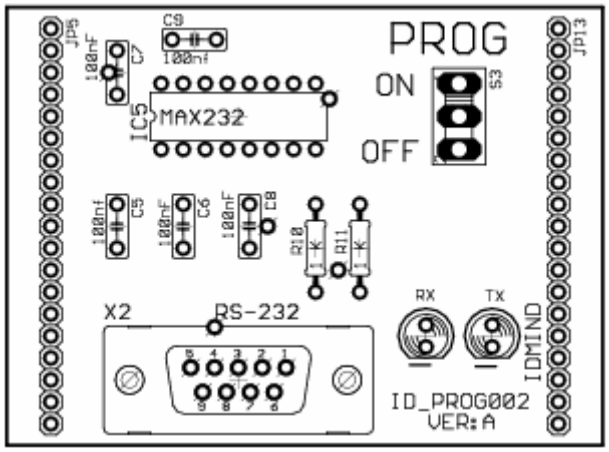
B3) Analog Interface Board - ID-AS002

Description	Quantity
4 pins male bars	2
19 pins male bars	2
Connection pins at 90º 3-pins	5
16 pins Socket	1
74HC4051N	1



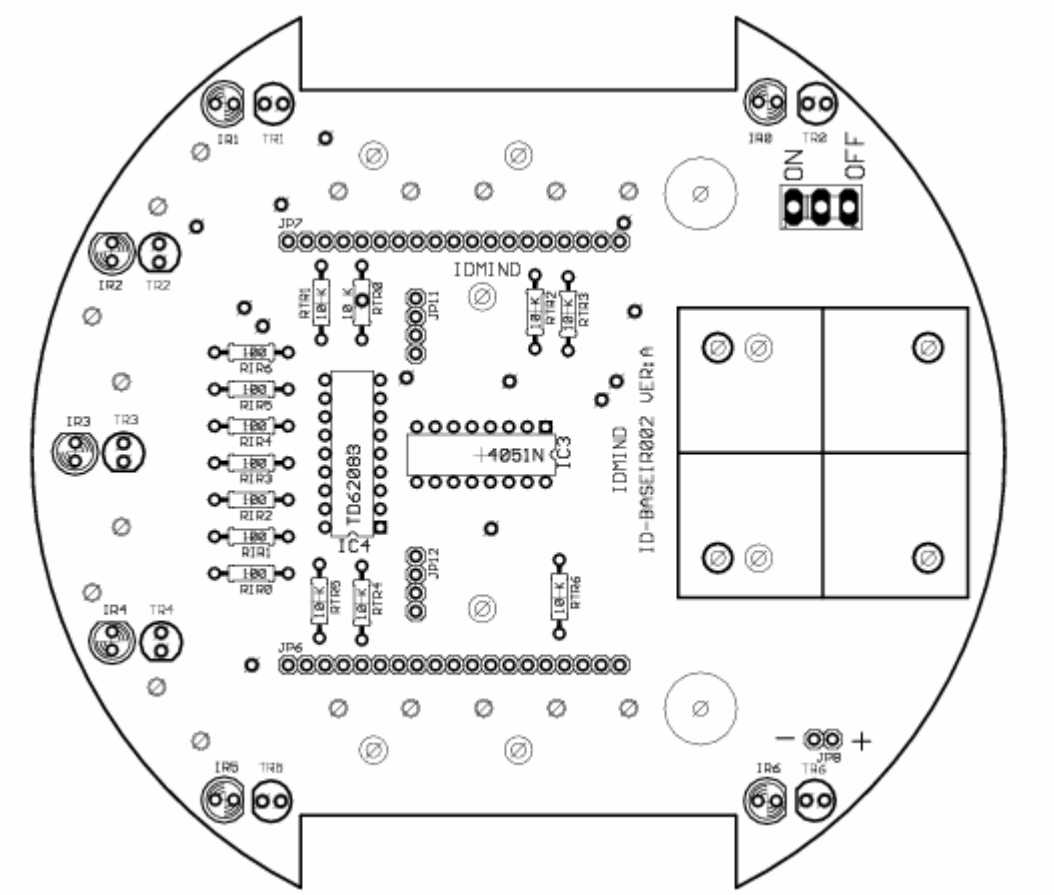
B4) Programming Board ID-PROG002

Description	Quantity
19 pins male bars	2
19 pins female bars	2
1K Resistors	2
Leds	2
100nF Capacitors	5
Interruptor	1
RS-232 to IDC	1
16 pins Socket	1
Max 232	1



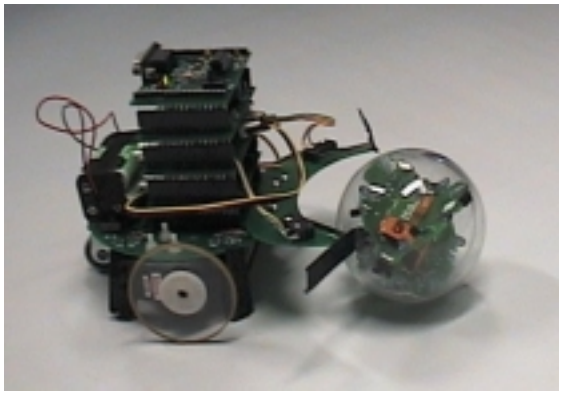
B5) Infrared Sensors Board ID-BASEIR002

Description	Quantity
Infrared Emitters	7
Infrared Receivers	7
100 Resistor	7
10 K Resistor	7
uln2803A Buffer	1
74HC4051N	1
18 pins Socket	1
16 pins Socket	1
4 pins female bars	2
19 pins female bars	2
Interlocking 16A 2 openings	1
Batteries 4*AA Support	1
Interruptor	1
Connection pins 2-pins	1
Contact Interruptors	2


IR1
TR1
IR0
TR0
ON
OFF
JP7
IR2
TR2
IDMIND
RTR1
10-K
10-K
RTR0
JP11
10-K
RTR2
10-K
RTR3
IDMIND
ID-BASEIR002 VER:A
IR3
TR3
RIR0
RIR1
RIR2
RIR3
RIR4
RIR5
RIR6
TD62003
IC4
4051N
IC3
JP12
10-K
RTR5
10-K
RTR4
10-K
RTR6
IR4
TR4
JP6
IR5
TR5
IR6
TR6
JP8
−
+

Appendix C– Circular Robot alterations to allow it playing soccer

e-mail: info@idmind.pt
http://www.idmind.pt



Soccer ROBOT

Assembly Instructions

V 2.0

April 2003

e-mail: info@idmind.pt
http://www.idmind.pt

Table of Contents

e-mail: info@idmind.pt
http://www.idmind.pt

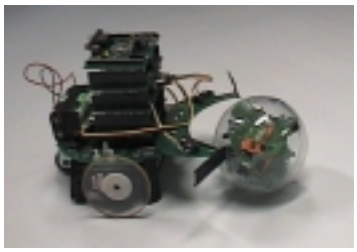
1 Soccer Robot Assembly

1.1 Introduction

This manual explains the assemble process for the soccer robot based on IdMind' *Circular Robot*.

Using the *Circular Robot* kit, the construction of a soccer robot develops in two different phases: a) the base structure assembly; b) specific aspects of assembling this type of robot.

The next subsections show the development of those phases.

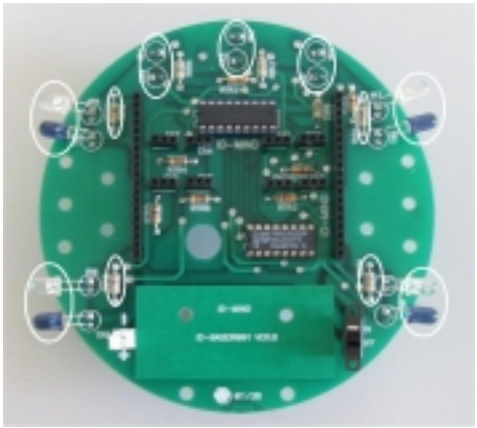


The above figure shows the final aspect of a soccer robot, and an infrared (IR) ball used for robotic soccer matches.

1.2 Base Structure Assembly

When following the assembly instructions of the *Circular Robot* kit, notice that the marked components in the figure must not be assembled:

Reference	Component
RIR 0, 1, 5, 6	100Ω Resistors
IR 0, 1, 2, 3, 4, 5, 6	InfraRed Transmitters
TR 0, 1, 5, 6	InfraRed Receivers
	Obstacle detection Micro-switches



The infrared detectors **TR 2**, **TR 3** e **TR 4** must be assembled with length **1 cm**, which is inferior to the recommendation for the *Circular Robot*.

After the IR detector assembly is finished, the assembly of the specific elements of the Soccer Robot must be carried out. These elements were chosen to meet the rules of a soccer Robot competition (*RoboCup Junior*).

1.3 Assembly of Specific Elements

1.3.1 LDR

One way to locate the robots inside the pitch is to use the marks on the pitch surface. For this purpose, the LDRs must be assembled, aiming at the measurement of the surface reflected light. This way, it is possible to detect the different colours (different grey levels).

Since infrared light emission it is not allowed in a soccer robot competition, the surface is illuminated using the light emitted by a green led, assembled in parallel with the detector led. In order to obtain a larger emission power, the emitter current must be increased. As such, it is necessary to lower the value of the resistor in series with the light emitters (RIR 0, 1, 5, 6), from 100Ω to 33Ω.

Thus, the components that must be assembled in this section are:

Reference	Assembled Component
RIR 0, 1, 5, 6	33Ω Resistors
IR 0, 1, 5, 6	Green Leds (high intensity)
TR 0, 1, 5, 6	LDR

Be careful with the green leds polarity. The LDRs do not have polarity, and they must be assembled to be as high as the leds light emitters, as show in the figure on the right (about 3 cm from the board to the top of the leds). A device such as an opaque tube must be placed in front of the leds so as for both of them to become more directional, and for the receiver not to get lateral light.

The figure below shows two of the four resistors that must be replaced. They are located on the backside of the robot (white arrows).

Each emitter/receiver pair must be covered by an opaque curtain so as to decrease the external light interference, isolating these elements from the external environment. For the covering we suggest the use of a black bristol board (3,5 cm X 3,5 cm), with a tube shape and covered with an isolating tape. The tube must have sides 1 cm and 0,5 cm.

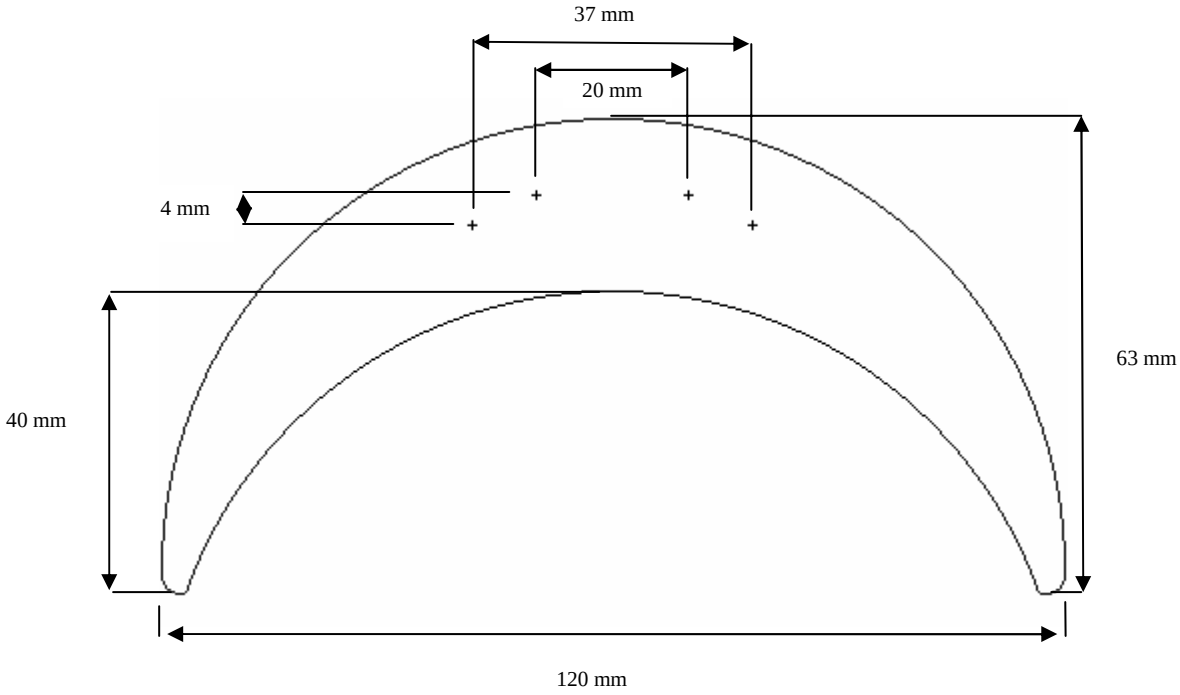
The curtain must be glued to the servomotor using a heat fuse glue, as shown in the figure below (the bristol board is glued to one servomotor):

Once the curtain and the opaque tube are assembled, the latter must be assembled with isolating tape on TR2, TR3 and TR4 sensors. After the tape is placed, the sensors must be folded so as to be parallel to the floor. They must point onward as shown in the following figure:

1.3.2 Half-Moon

One of the fundamental requests for a soccer match is to keep ball control. Ideally, the robot must be able to keep the ball always in its front. We recommend the installation, in the robot's front, of an "half-moon" shaped device parallel to the floor. This will allow ball control in a safe way, obtaining good control over its directionality.

The next figure shows a suggestion for the half-moon shaped device. The device's final shape can be any other shape.



The holes used to fix the device (+) must be according with the marks of the robot base. The device must be fastened to the robot base with screws, as shown in the figure below. The half-moon shaped device will replace the contact sensors (used by the circular robot) assembly location.

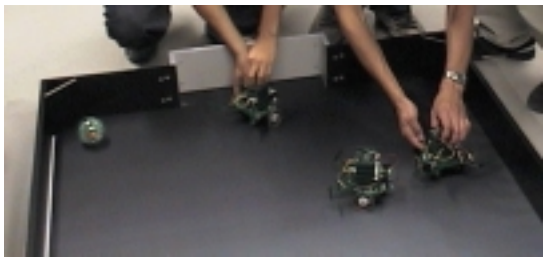
1.3.3 Contact Sensors

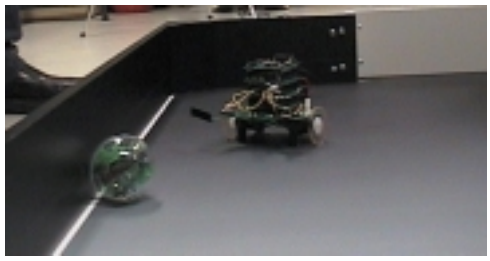
The contact sensors of the *Circular Robot* are replaced, on the soccer robot, by the half-moon shaped device that allows ball control. They are available to be placed in another spot where they can be used, e.g., to detect collisions.

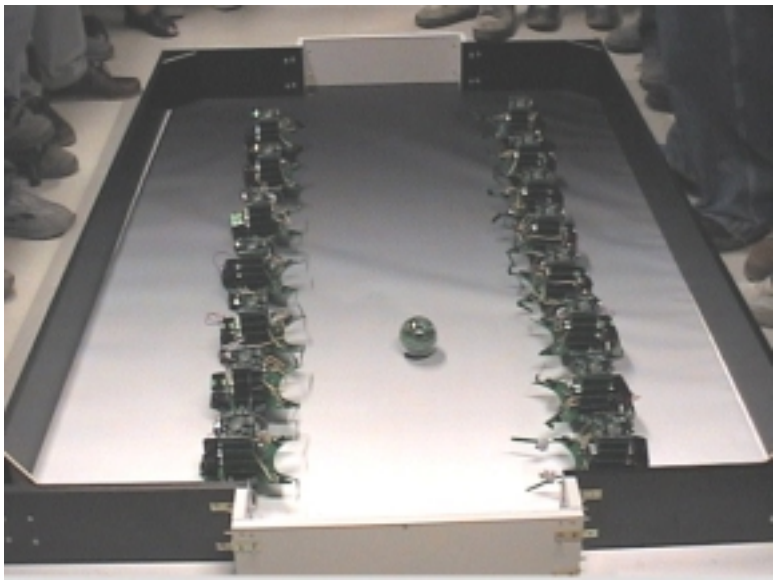
We recommend these sensors to be placed where they can detect undesirable collisions, providing the information to the controller, to solve conflict situations during a game. A good place for its assembly is the half-moon shaped device itself.

1.4 Examples of Applications

The figures below show several competitions and events where IdMind' Soccer Robots where used:







Good Games!!!

IdMind' contacts:

IdMind - Engenharia de Sistemas, Lda
Centro de Incubação e Desenvolvimento - CID
Pólo Tecnológico de Lisboa, Lote1
1600-546 Lisboa
PORTUGAL

Tel: +351 21 7101100
Fax: +351 21 7101103
http://www.idmind.pt
e-mail: info@idmind.pt