Lebanese American University

School of Engineering

Embedded Systems

COE 521

Project – Robotic Pet

Submitted to: Dr. Zahi Nakad

Prepared by:
Joseph Attieh 201501102
Clara Akiki 201405063
Mia Antoun 201502038

December 23rd, 2019

Table of Contents

Table of Figures

Table of Tables

Introduction

In this project, we are asked to use the Robot 5 platform in order to mimic the movements of a pet. There will be two main features in this project; The first feature is the “Keep distance” in which the robot must keep a fixed distance of 20cm between the ultrasonic sensor and a certain object. The second feature is the “Fetch” in which the robot must fetch a ball and bring it back to initial position.

Part 1: Design & Components

The first part of this pet project is to implement the "Keep the distance" aspect. The rover should keep a distance of 20 cm from an object. If there is no object, the rover should stop moving. In order to perform this task, we will use an ***LCD*** for debugging purposes (it will display the distance between the rover and the object), an ***Ultrasonic Sensor*** for calculating the distance between the rover and the object and a ***Motor Driver*** in order to manipulate the movements of the rover. In this part of the project, we will only use the ***PIC18F4550***.

PIC18F4550

PIC18F4550 is an 8-bit microcontroller member of the PIC18F family. In order to program the pic, we used Pickit3 alongside MPLAB X IDE.

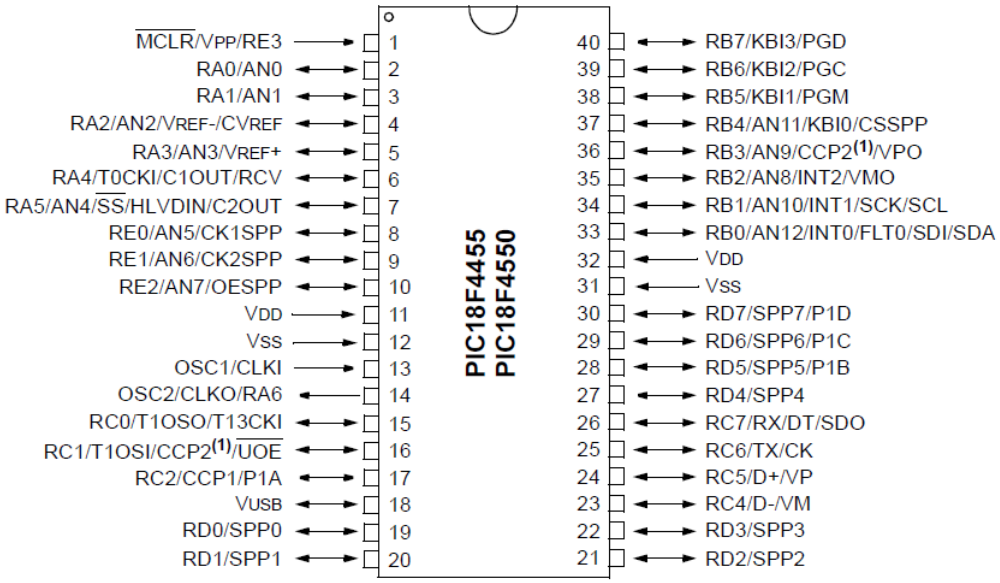


Figure 1: Pin Diagram of PIC18F4550 taken from the datasheet

Port A (pin RA0) will be used for the analog outputted by the ultrasonic sensor.

Port B will be used by the motor drivers.

Port D will be used for the data of the LCD and Port E will be used for the RS and EN of the LCD.

Voltage Regulator

In order to prevent damaging the processor inside the PIC, we should supply a voltage of 5V. However, the battery that we used to power the Bidirectional motor driver is 9V. For this reason, we will use a voltage regulator. Its input is connected to the battery while its output is connected

to the PIC, ENA and ENB.

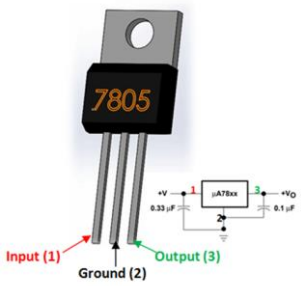


Figure 2: Voltage regulator pins and circuit extracted from: https://components101.com/7805-voltage-regulator-ic-pinout-datasheet

16x2 LCD Module

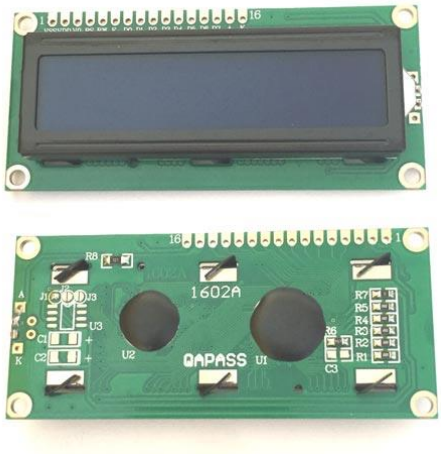


Figure 3:Picture of the LCD module taken from the following link: https://components101.com/16x2-lcd-pinout-datasheet

The LCD used is a 16×2 LCD. It has two registers: a command register and a data register. The user is capable of selecting the register by changing the value of RS.

The LCD has 16 pins and each pin is connected to a specific part of our microcontroller.

- Pin 1 is connected to the ground of our system.
- Pin 2 is connected to the circuit's power source (the output of the voltage regulator used).
- Pin 4 is the register select RS and is connected to pin 0 of PORTE. It selects command register when low, and data register when high.
- Pin 6 is the enable EN and is connected to pin 1 of PORTE. It sends data to data pins when a high to low pulse is given (virtual clock).
- Pin 7 to Pin 14 are the data pins DB0 to DB7 and are connected to PORTD.

Ultrasonic

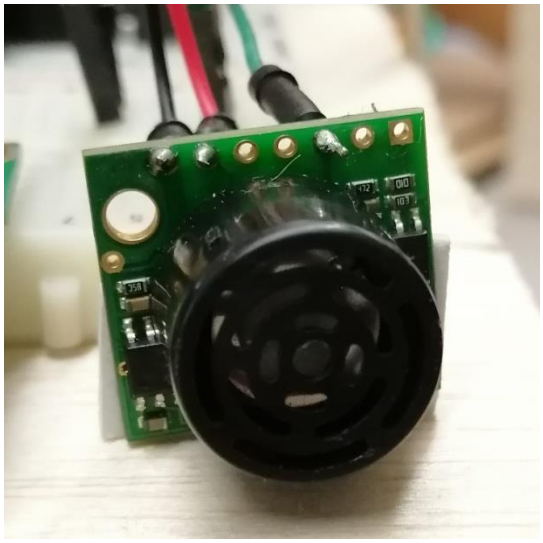


Figure 4:LV-MaxSonar

The LV-MaxSonar is an ultrasonic sensor that detects objects within a specific area. This sensor is not affected by the color or other visual characteristics of the detected object. High frequency sounds are used to detect and localize objects in a variety of environments.

The sensor has 7 pins and in our project only three pins are used.

- Pin 3 - AN: This pin outputs a voltage that is proportional to the distance. This analog voltage is connected to pin 2 RA0/AN0 of the PIC
- Pin 6 - Vcc: This pin operates on 2.5V - 5.5V and is connected to the V_{DD} of the PIC
- Pin 7 - GND: This pin is connected to the Vss pin of the PIC.

In our project, we should keep a fixed distance of 20 cm between the rover and the object in front of it. The object can move away or towards the rover and the 20cm distance should be kept. If there is no object in range of the sensors when this mode starts, the rover should just stop.

If the object is at a distance less than 19 cm from the rover, the rover should move backward until it reaches the allowed distance from the object. On the other hand, if the object is at a distance greater than 25 cm from the rover, the rover should move forward until it reaches the allowed distance from the object. This range was found as the best range for the rover to stabilize as the readings tend to fluctuate for a precise distance of 20 cm. If no object is within the range of the rover, the rover stops.

For the sensor to detect the value in centimeters, we start by reading the analog value and we convert it into a digital value by using the analog to digital converter in the pic. PIC18F4550 has 13 channels which mean 13 analog input signals can be converted simultaneously using the module. It uses a successive approximation with a resolution of 10 bits. That means that the range of bits is 0 to $2^{10} - 1 = 1023$. Since the maximum voltage that we want to detect is 5V, this means that 1023 corresponds to 5V.

In order to do so, we configure the registers responsible for this conversion:

1- ADCON0 (A/D Control Register 0):

This register is used to select the input channels. The GO bit (bit 1) starts the conversion of the input analog signal when set and is also used to check the status of the A/D conversion. Since we have 13 channels, it would make sense to have 4 bits to select those channels. The bits CH3: CH0 bits (bits 5-2) are the channel select bits which are multiplexed with digital I/O pins.

2- ADCON1 (A/D control register 1):

The bits PCFG0: PCFG3 (bits 3-0) determine which pins of the controller should be configured as analog inputs while the bits VCGF1: VCGF0 (bits 5-4) are used to select the reference voltage for conversion.

3- ADCON2 (A/D Control Register 2):

The bits ADCS2: ADCS0 (bits 2-0) configure the prescaler used for the A/D clock. The bits ACQT2: ACQT0 (bits 5-3) are used to set the acquisition time of the ADC. The ADFM bit is used to select the format by which result should be stored in the ADRESH and ADRESL.

4- ADRESH & ADRESL: Those registers are used to store the result of the conversion (10 bits).

In order to find the analog voltage measured, we used the following formula:

$$\frac{\text{Maximum value outputed by the A/D}}{\text{Maximum voltage}} = \frac{\text{A/D Reading}}{\text{Analog Voltage Measured}}$$

In our case: $\frac{1023}{5} = \frac{\text{A/D Reading}}{\text{Analog Voltage Measured}}$

Since the output of the sensor is connected to the analog pin AN0, we use this formula to get the value of the analog voltage from the content of ADRESH and ADRESL. Once we get this value, we need to get the distance measured. The datasheet of the sensor indicates that a supply of 5V to the Ultrasonic sensor yields around 9.8mV per inch. Since 1 inch is 2.54 cm and the voltage obtained by the sensor is in V, we can calculate the distance using this formula:

$$distance = \frac{\text{Analog Voltage Measured} * 2.54 * 1000}{9.8}$$

The distance measured is used to choose whether the rover should stop, move forward or move backwards. It was also displayed on the LCD for debugging purposes.

Encoders

For the rover to be able to go forward, backward, left and right, we will need the encoders for the motors. Since we never worked with encoders before, we had to do an extensive research in order to know what they are and how to use them. This research was done before starting our implementation. Even though we chose not to use encoders (for time constraints and simplicity), we include this section for completeness.

What are encoders?

Encoders are components that are added to a DC motor engine to change the mechanical movement into digital pulses that can be deciphered by the integrated control electronics. The primary intention behind the different kinds of encoders is to transform data from one format to another for standardization, speed control or safety control. DC motors have a perplexing position and speed control, their behavior isn't linear and depends intensely on the load borne; it is the reason why they need encoders to determine and guarantee a right shaft position.

How do encoders work?

An encoder is made of a disk connected to a rotating shaft. This disk is a combination of transparent and opaque areas that block the passage of light for this disk to be coded. As for how the encoder is operated, whenever the shaft mentioned earlier rotates, the infrared source emits a light that is translated by a phototransistor (or optical) which generates the digital pulses depending on whether the light emitted passes through the disc or no is blocked by the opaque regions. We will therefore get an information sequence that allows for the control of aspects such as the direction of motion and turning radius that we will be using in our project.

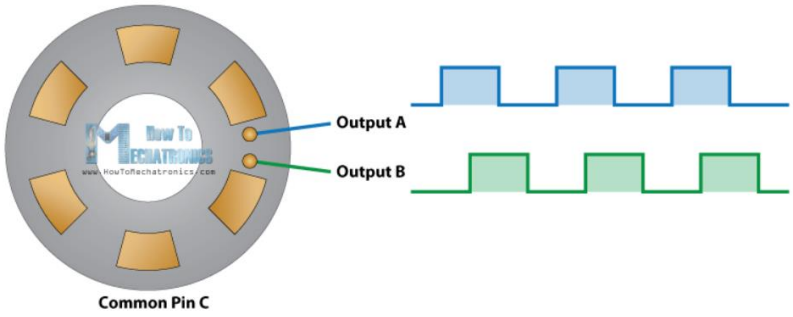


Figure 5: Encoder and the pulses it generates from https://howtomechatronics.com/tutorials/arduino/rotary-encoder-works-use-arduino/

After we had a greater understanding on the function of the encoders, we went even deeper to understand their function in the Rover 5. The type of encoders used are quadrature encoders whose resolution is 1000 state changes per 3 wheels rotations.

The encoders used in our rover are the following:

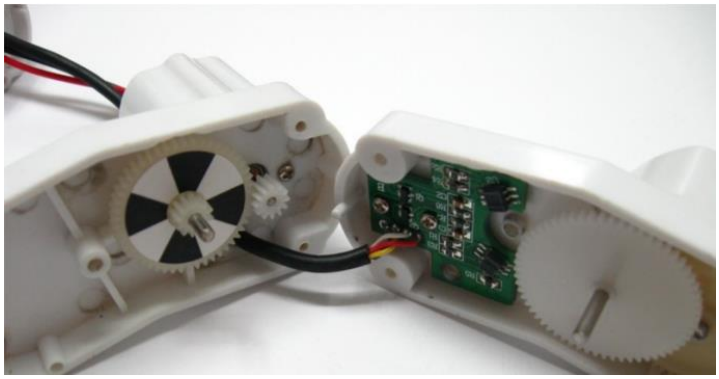


Figure 6: Encoders in Robot 5 from http://image.dfrobot.com/image/data/ROB0055/Rover%205%20Introduction.pdf

We can see that there are 4 wires:

- Red: Vdd = 5V
- Black: Vss = 0V (GROUND)
- White: Signal A
- Yellow: Signal B

Signals A and B determine the sense of direction of the encoder. As explained earlier, the sensors used will give us an output of 2 square waveforms 90degrees out of phase as show in the below image:

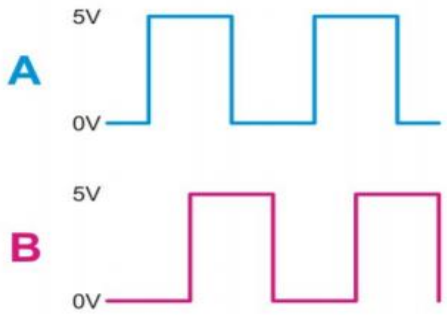


Figure 7: Square waveforms of signals A and B from http://image.dfrobot.com/image/data/ROB0055/Rover%205%20Introduction.pdf

The 2 outputs are used to get the speed of rotation and the direction of shaft rotation by looking at the binary numbers generated by both outputs. Since we are only concerned with the speed of rotation then we only need to connect one of the wires and measure the frequency.

We learned how to use encoders in order for the rover to move but we found that the simplest way to proceed is by not using them.

Motor Driver UK1122

In order to implement this design, we used UK1122 which is an H-Bridge Dual Bidirectional Motor Driver. The motors A and B will supply power to the Rover's motors. The module receives its power from a 9V battery, and it has the following pins: ENA, ENB, IN1, IN2, IN3 and IN4. Both the enables turn on two pins on each side of the driver and these + and – pins are connected to each motor. Whenever ENA is high and ENB is high, we would be enabling the pins that control the motor A and motor B accordingly. IN1 and IN2 are associated with ENA and IN3 and IN4 are associated with ENB; they allow us to determine the direction of the two motors of the rover. Furthermore, they allow us to control the movements of the rover: forward, backward, left, right or stop.

Figure 8:https://www.rpelectronics.com/uk1122-l298-h-bridge-dual-bidirectional-motor-driver.html

Rover 5

The bidirectional motor driver will send voltage to the motors of our rover. We connected each two consecutive motor on the same side of the rover together because we want the two wheels of each side to work together. Each two motors will be controlled independently by 'MotorA' and 'MotorB' of the UK1122 motor driver. Whenever IN1=1 and IN3=1, both the motor of the same side of the rover have to go forward.

On the pic side, we connected IN1, IN2, IN3 and IN4 to RB0,RB1, RB2 and RB3 respectively.

In previous labs, we wrote methods that we will use to control the rover. The table below depicts the logic of the code:

Table 1: Logic of the motor driver

Direction	In0	In1	Right Motors	In2	In3	Left Motors
Forward	1	0	forward	1	1	forward
Backward	0	1	backward	0	1	backward
Left	1	0	forward	0	1	backward
Right	0	1	backward	1	0	forward
Stop	0	0	stops	0	0	stops

Part 2: Design & Components

The second part of this pet project is to implement the "Go Fetch" aspect. The rover should look for a green ball, go fetch it and bring it back to its initial position. In this part of the project, we will use the ***Raspberry Pi*** and the ***Pi camera*** in order to act as the eyes and brain of the system by detecting the ball and sending commands to the ***PIC18F4550*** to move the rover (move right, left, forward, backward or stop).

Raspberry Pi

The pi is powered through an external power supply. That same power supply feeds the Motor Driver and the voltage regulator powering the PIC. We used VNC Viewer to have a graphical remote access to our Pi. The main aim of the Pi is to detect the green ball by performing image processing and sending commands to the PIC through its GPIO pins. In order to identify the ball, we used the Pi Camera and OpenCV (code can be found in the below section). In order for the Pi to send instructions to the PIC (move left, right, forward, backward or stop depending on the position of the balloon), we used three GPIO pins set up as output (pins 36, 38 and 40). The description of both of these implementations can be found in the following sections.

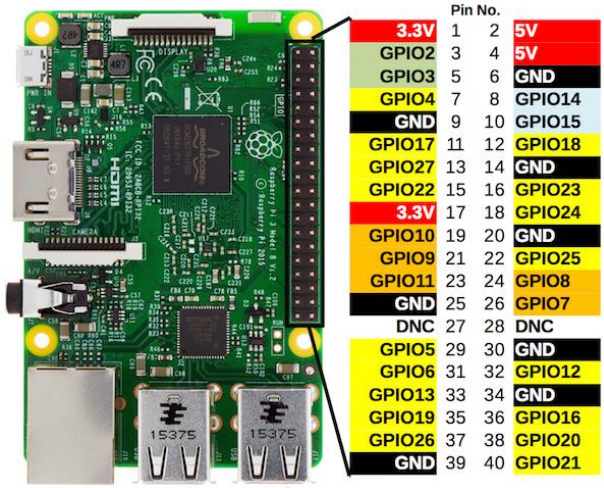


Figure 9: Raspberry Pi pin diagram taken from: https://components101.com/16x2-lcd-pinout-datasheet

Figure 10: Pi Camera connected to RPi taken from: https://components101.com/16x2-lcd-pinout-datasheet

Vision

The raspberry pi is responsible of detecting the ball and sending commands to the PIC. The python script implemented takes pictures using the PI Camera and uses OpenCV to detect contours of clusters of green colors in the image. The largest area of green color represents our green ball. We used HSV instead of RGB to detect the green color with the following ranges for the green color:

$$min\_green = np.array([60 - 25, 20, 20])$$

$$max\_green = np.array([60 + 25, 255, 255])$$

Then, the center and the area of the ball are computed using OpenCV. The center of the detected green entity is used to determine whether it is on the left, right or mid part of the picture taken (vision of the camera). This determine the direction of motion that the rover should be taking to reach it. The area of the detected green entity will be used to check whether it is too far or too close from the camera. This usually means that the ball is now fetched which should be initiating the backtracking process.

After detecting the ball, we need to send instructions to the PIC for controlling the movements of the rover to move it towards the ball. We examined different ways to relay information from RPi to PIC and we ended up choosing to send the data over the GPIO pins of the PI. We used three pins (36, 38 and 40) set as outputs to determine the movement of the rover. Pin 40 is mainly responsible for determining whether the rover fetched the ball or not and initiates the backtracking process. The table below depicts the values and their meaning:

Bit 2 (pin 40)	**Bit 1 (pin 38)**	**Bit 0 (pin 36)**	**Output Instruction(s)**
0	0	0	Stop
0	0	1	Forward
0	1	0	Left
0	1	1	Right
1	0	0	Backward

Table 2: Logic of the RPI/PIC connection

However, the GPIOs of the RPi output a voltage of 3.3 V while the PIC operates at 5V. In order to solve this problem, instead of using a level shifter, we decided to perform an Analog to Digital Conversion to read the voltage received by the PIC. Thus, the PIC converts the value received on its analog channel and checks whether this value is greater than 3V which represents a logical 1. Otherwise, the value detected is a logical 0. We set the pins as follow:

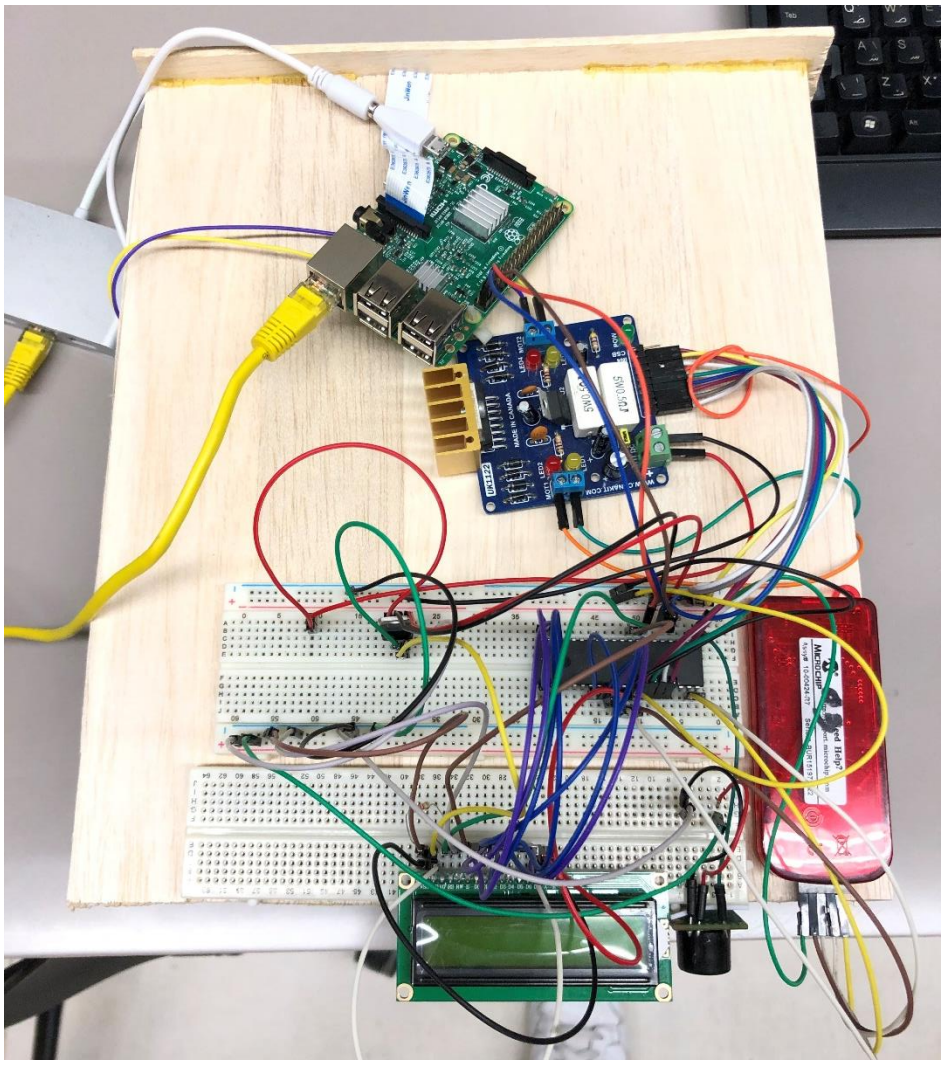
- Pin 36 of the RPi will be connected to pin 7 (AN4) of the PIC.
- Pin 38 of the RPi will be connected to pin 5 (AN3) of the PIC.
- Pin 40 of the RPi will be connected to pin 4 (AN2) of the PIC.

In a nutshell, the RPI takes a picture of its surroundings using its camera. After detecting the ball, the PI checks the position of the ball in the picture by subtracting the center of the ball and the center of the picture. Based on the value of the difference, the PI sets the values of pins 36,38 and 40 to command the pic to move towards the ball (left, right or forward). The command is saved in a list in order to initiate a backtracking sequencing once the ball is fetched. The PI knows that it reached the ball once the radius of the area detected is higher than a certain threshold (the ball is very close to the camera). Once this condition is satisfied, the RPI has fetched the ball and should return to its initial position. This is done by reversing the list of commands that were sent and reversing their function (the instruction that was previously forward is now sent as being backward and so on). This sequence is optimized before proceeding with the backtracking. The optimization is done by counting the number of left turns and right turns and subtracting those counts (each turn left eliminates a turn right and vice versa). After getting the optimal list, the commands are sent from the RPI to the PIC.

On its end, the PIC reads the value of the voltages at AN2, AN3 and AN4 and checks whether the values represent 0s or 1s. Based on the sequence received, the PIC sends commands to the motor driver to move in a specific direction. The PIC is unaware of the status of the ball; it just performs the instructions received (toggles IN1, IN2, IN3 and IN4 to change the direction).

Part 3: Picture of the Board

This is the picture of the circuit used for both parts of the project:



Conclusion

This final pet project mimicked a robot that acts as a pet. Two main aspects were implemented: “Keep distance” and “Fetch”. It was divided into two separate parts: first one is keeping a distance of 20cm between the rover and an object; second one is fetching a green ball and bringing it back to its initial position. This project used the Raspberry Pi, the PI camera, the PIC18F4550 and the ultrasonic sensor.

References:

https://clr.es/blog/en/types-of-encoders-and-their-applications-in-relation-to-motors/

https://howtomechatronics.com/tutorials/arduino/rotary-encoder-works-use-arduino/

https://clr.es/blog/en/types-of-encoders-and-their-applications-in-relation-to-motors/

http://image.dfrobot.com/image/data/ROB0055/Rover%205%20Introduction.pdf

https://www.maxbotix.com/documents/LV-MaxSonar-EZ_Datasheet.pdf

http://www.raspberrypirobotics.com/raspberry-pi-gpio-access/

https://components101.com/16x2-lcd-pinout-datasheet

https://projects.raspberrypi.org/en/projects/getting-started-with-picamera