ENGR 1190 -Test Document

EV3 Pick and Place Robot

Submitted to:

Paul Grand

By:

Carlos Vasquez

Kevin Vo

Douglas College

August 3, 2021

**ENGR 1190 – TEST DOCUMENT**

The main idea of this test document is to describe the test strategy, objective, software testing scope and activities of our EV3 Pick and Place Robot. For which we can determine the effort needed to validate the quality of the application under test that we minutely monitored and controlled to have a better result for the robot. In this case, we have decided to test every single sensor, record different data values that will help us in the final design of the robot. Additionally, the test document is designed in a way that can provide information about the functionality of the code that can guide students to create similar projects in the future.

**What do we what to test:**

During the process of creation, we plan to use four different sensors and two motors that will help the robot to function ideally. To ensure the sensors work properly we divided the software testing into 4 levels:

* Unit testing
* Integration testing
* System testing
* Acceptance testing

Each sensor will have its level of testing to validate that each unit of the software performs as designed. Additionally, the 4 levels will expose any faults in the interaction between integrated units. Test drivers and test stubs are used to assist the project idea.

**Sensors tests before the final project:**

**Gyro sensor:** Device containing a rapidly spinning wheel or circulating beam of light that is used to detect the deviation of an object from its desired orientation.

* Software testing levels:

To ensure that the gyro sensor was working properly we measure the accuracy of the gyro. We had design another EV3 robot that turned a known angle to the left and later to the right. We decided to have this known angle to be 90° and base on the different tries we concluded that the accuracy of the clockwise and counterclockwise turns we almost the same. This also agreed with the ± 3-degree accuracy that the manufacturer claimed.

**Touch sensor:** This is an analog sensor that detects physical objects when its red button gets pressed, then the pressure will make it send an analog signal.

* Software testing levels:

For this particular sensor, we designed a program that uses two touches sensor. One sensor that when is pressed would change the LED to red and if the other sensor is pressed would change the LED to green. Additionally, we created another program that could count the number of times the sensor is been pressed. This helped us verified the accuracy of the touch sensor.

**Ultrasonic sensor:** This is a digital sensor that can measure the distance to an object in front of it. It does this by sending out high-frequency sound waves and measuring how long it takes the sound to reflect back to the sensor. The distance can be measured in either inches or centimetres.

* Software testing levels:

To make sure the ultrasonic sensor works, we designed a program that measured (in cm) the distance of a piece of paper that was positioned in 5 different random points. We then collected the data and confirmed the ± 1-centimetre accuracy that the manufacturer claimed. We also did the same test using inches and we got an accuracy value of ± 0.395 inches.

**Colour sensor:** This is a digital sensor that can detect the colour or intensity of light that enters the small window on the face of the sensor. The light sensor takes in the light, and like the colour sensor, takes in the ambient light in the range of 0 to 100 through the phototransistors.

* Software testing levels:

To ensure that that sensor colour was working properly, we had to design a program that can detect the colour of a tape that was positioned in front of the sensor. For this test, we had chosen 6 different colours: black, blue, yellow, red, green, and white. After testing several times the program the sensor identifies all single colours without any problem. However for best accuracy, when in colour mode or reflected light intensity mode, we found that the sensor must be held at a right angle, close to the surface it is examining and avoid touching it to have better results.

**Software design for the final project :**

After we individually tested all the sensors and applied different software testing levels, we were confident that we could now implement all the sensors in one project. As a result, we have decided to design the creative Pick and Place Robot and use C language to code our robot.

**Coding points**

*#define*

To make our code to be easy to adapt to different hardware configurations, we defined every single port that each sensor would use.

Table

Description automatically generated

Variable a data type used:

Text

Description automatically generated

For this particular program, we used 5 different functions that are easy to follow and to understand.

* Grab function   
  Text

  Description automatically generated

We set the motor to reversed at a power speed of 20% for the grabber to pick up the object because of the way the motor is mounted. As the object is to grab the delay(500) will stop the motor for 0.5 and eventually stop and wait again for 0.5 seconds to finish the first function and display on the screen “ Pick up”.

* *To consider*: *In order to figure out the speed of the grabber we performed different speed tests and we conclude that the best power speed was between 19%-21% and after calculating the average 20% was the best speed number*.
* Drop function

Text

Description automatically generated

Similar to the previous function this function what it does is set the motor to go forward instead or reversed it. The speed-power at which the grabber will release the object is 8% and once the object is released a one-second time lap will be executed and then stop the motor. Following that the motor stops completely and waits for one more second to finish the function and display on the screen “Release”.

*To consider: As you noticed the releasing/ dropping time is different from the grab function. This is because when we were testing the robot several issues were encounter if we use less time. Also, the speed-power was set to be 8% because as the object is dropped we wanted to fall slowly rather than fast and break.*

* Within reach function

Graphical user interface, text, application

Description automatically generated

This function is a boolean value that returns true or false. It receives a float distance value for the ultrasonic sensor to read the distance at that specific moment. The sensor makes sure that the distance between the sensor and the object is a range of 0 and 5 cm. If something is in this range the object will be picked up.

*To consider: We set up the distances to be between 0 and 5 cm due to a test that we did to ensure that the distance from the ultrasonic sensor and arm were correct. With the test, we could calculate the total distance of the arm and subtract the difference between the sensor and the grabber for the object to be picked up.*

* Detected function.

Graphical user interface, text

Description automatically generated

This is another Boolean value that receives long values of degrees between 0 to -90°. The time the motors stop depends on where an object is positioned for the given range. We have divided the range of rotation into 3 different sections for the robot to detect the object.

1. -20° to -40°
2. -40° to -60°
3. -60° to -90°

*To consider: If you noticed the delay time increases as we get farther away from the initial point (0°), this is because as we were testing, we realized that as the crane moves the moment of inertia that it has increases and more time needs to be used to stabilize the crane and detect the object.*

* Set mode function

Text

Description automatically generated with low confidence

The way this function works is by using the colour sensor to identify 3 different colours that have the following functionality inside if loops:

* Red = 5; (shut down the whole program)
* White = 6; (Pauses the program and looks for either green or red)
* Green = 3; (Resume the program)

*To consider: We were planning to use yellow instead of white, but we found out that during several tests the colour sensor was not recognizing yellow and in other to avoid any problems we decided to use white.*

* The main function

Text

Description automatically generated In the main function, we can see how we have the first while loops that run the whole program and inside that there is another while loop that starts the initial calibration/position and sets the motor to 30% speed-power to swing the arm back to the right until the touch button gets pressed. Once the button is pressed the program will stop the motor and reset the gyro. We also have the *drop()* function here because we always want to have the gripper open at the beginning and then break the second while loop if the conditions are met.

* Continuation of the main function

Graphical user interface, text, application, email

Description automatically generated

Now the third while loop takes place here and grabs the degree from the gyro to check if the boolean conditions are met to continue running this part of the code. Therefore, the withinReach(getUSDistance (Ultra))) is check to ensure that the conditions from this function are either true or false. If true execute as usual, but if it is false break the loop and move on.

Graphical user interface, text, application

Description automatically generated

In this part of the code, if the degree value is bigger or equal to -80°, the speed motor will rotate backwards at 10% speed-power. Also if the degree value is less or equal to -80° the motor will stop and break the loop.

*To consider: The setMode() function has been positioned in crucial areas where we have to check the colour without bugging the code. Additionally, we have programmed the robot in a way that is smart enough to pick up a different object and move the previous one and positioned the new object to the final checkpoint without affecting the program as shown in figure 1.*

Diagram

Description automatically generated

*Figure 1. The robot picks up a different object and moves the previous one.*