Visual Programming, Robotics and Augmented Reality in Introductory Programming course

by

Rahul R (MT2013119)

A thesis submitted

in partial fulfilment of the

requirements for the degree of

Master of Technology

in

Information Technology

International Institute of Information Technology, Bangalore.

June 2015

Thesis Certificate

This is to certify that the thesis titled **Visual Programming, Robotics and Augmented Reality in Introductory Programming course** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Technology** is a bona fide record of the research work done by **Rahul R (MT2013119)** under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Madhav Rao

IIIT-Bangalore,

The 18th of June, 2015.

Abstract

"In this research process, attempt is to build a new interactive educational-product for students so as to inspire their interest in learning programming. The research objective in this thesis is an educational robot for students which can interact through visual programming. Two versions of the hardware platform is made available, one a ready made platform known as the Electric Ray Robot and another an easily buildable robot, known as the Inventor's Bot or InBot, thus giving a hands on experience in introductory robotics as well. A visual programming environment is also developed for desktop (MiniBloq) and mobile (InBlocks) devices to program the robots. An augmented reality simulation engine is integrated to the programming environment in a view to reach out to children with limited access to the hardware platforms. The results show that there is a promising and huge market potential to apply the new interactive technology to the development of educational robots."

Acknowledgements

This work might never have been completed without the generous help and support of my supervisor, Prof. Madhav Rao, who was always eager to help and his door was always open for discussion. I would like to thank him for being my mentor during the past six months and for all those long discussions that we had. I thank him for introducing me to the field of visual languages and most importantly for teaching me how to convey ideas in writing.

I would also like to thank Prof. Jaya Sreevalsan Nair and Mr. Ramesh Sundararaman for agreeing to be a member for my guiding committee.

Last, and not the least, I would like to thank my family who have always supported me. Without their support, encouragement and love this work would have never been completed.

— Rahul R

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A programming curriculum in an introductory programming syllabus requires students to learn foundational programming skills and concepts. Conventionally a programming language is taught with a Hello World example. Consider a simple problem to display Hello World on the console. Following are the programs written in three popular programming languages: C, Java and Python, as shown in the listing 1.1, 1.2 and 1.3.

Listing 1.1: Hello World program in C

```
#include <stdio.h>
int main()
{
    printf("Hello World!");
    return 0;
}
```

Listing 1.2: Hello World program in Java

```
public class HelloWorld {

    public static void main(String[] args) {

        System.out.println("Hello World!");

    }

}
```

Listing 1.3: Hello World program in Python

```
print "Hello World!"
```

Each programming language presents its own set of difficulties to make it understand for a beginner. In C, the challenge for students is to understand the concepts of main, preprocessor directive, header file and return type in a first instance. The Java program too involves concepts of class, access modifier, static keyword and method call. The Python program is more intuitive than C or Java. However the novice programmer has to remember the function to print the text. It was felt that the beginners tend to learn the intricacies of programming if they are motivated. The idea is to introduce to a set of building block functions which is easy to program and then introduce students with similar concepts in any of the programming languages mentioned above.

In this visual approach to programming, a programmer need not remember any of the constructs of a programming language, but only on the approach to solve a problem. Visual programming has been an active research area in recent years resulting in many visual programming languages. In computing, a visual programming language (VPL) is any programming language that lets users create programs by manipulating

program elements graphically rather than by specifying them textually. Scratch 1.1 developed by the MIT Media Labs in 2005 is a popular visual programming language. Scratch is used by students, scholars, teachers, and parents to provide a stepping stone to the more advanced world of computer programming.

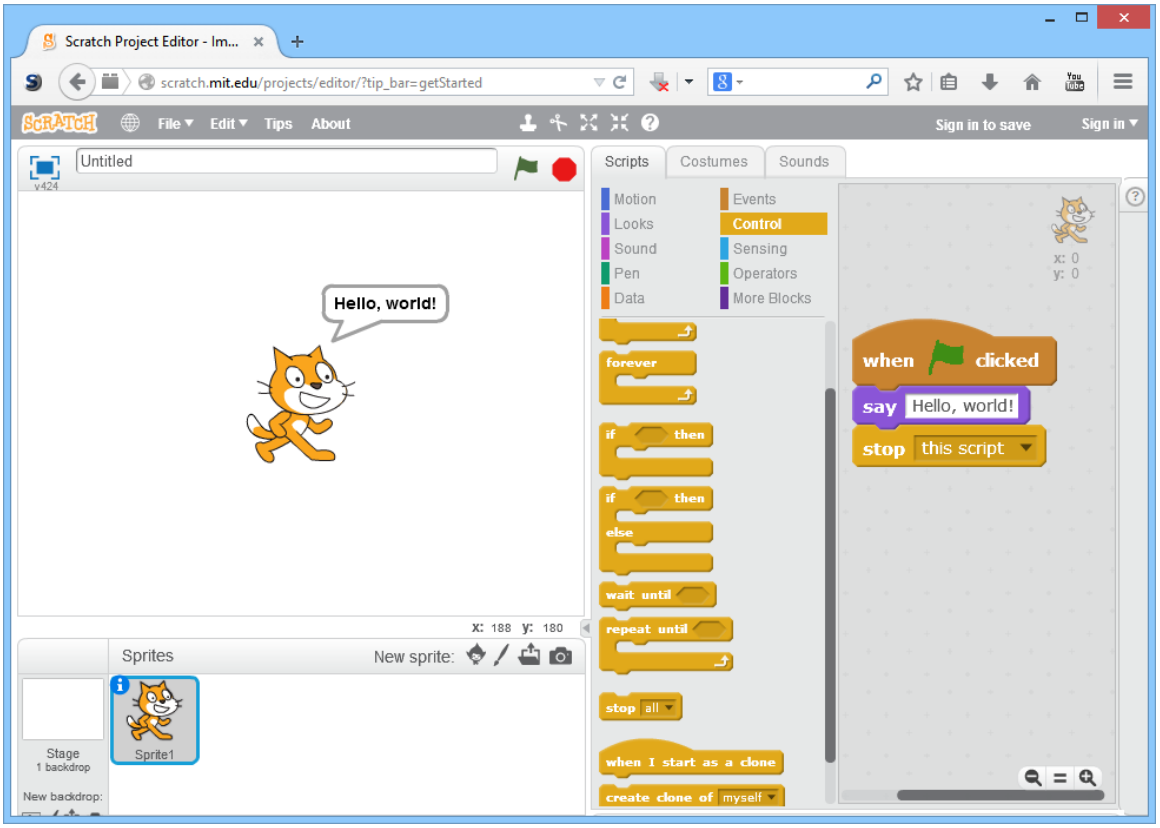

Figure 1.1: Hello World program in Scratch

In this research, we extend the idea of using visual programming language with a real robot. This gives a platform to learn fundamentals of programming as well as robotics.

LEGO is known for its educational robotics line through the Mindstorms robotics kit since 2006. The Education EV3 Core Set consists of EV3 programmable brick

(ARM9-based processor), motors, different types of sensors and around 500 accessories. Very simple programs for the robot can be created using the programmable brick itself. For complex programs, a visual programming software is included in the package. Lego's Mindstorm kit was adopted by Department of Computer Science of US Air Force Academy [2] to teach programming. However the LEGO kits are not affordable and the building blocks developed is not based on open source designs. The open source design implies that both hardware and software can be independently extended if required. A similar work was done by University of Alabama Computer Science Department which used a 3D graphical environment Alice [6] and iRobots for its introductory CS course [1]. Their course topics cover objects, methods, variables, loops, nested ifs, user input, parameters, events, random numbers, arrays and recursion. The software development was made open source, yet the hardware design was restricted to iRobot proprietary design.

Hence a need for an economical robotic kit with an open source hardware design and software platform is required for the beginners. The subsequent chapters disusses in detail the design and development of this interactive educational platform.

Chapter 2

Electric Ray Robot

Electric Ray robot developed by Protocentral Inc, as shown in Figure 2.1, is an Arduino compatible [**?**] robot. It is an entry-level, low-cost complete robotic platform. It is simple enough to be used out-of-the-box for teaching and learning robotics and Arduino programming.

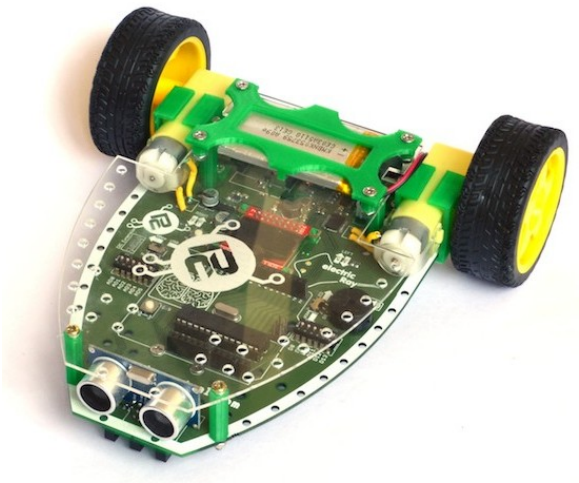

Figure 2.1: Electric Ray Robot

2.1 Specification Summary

- Atmega328 microcontroller with Optiboot bootloader (Arduino-Uno compatible)
- Arduino shield expansion headers
- Ultrasonic distance sensor
- 3x IR sensors
- OLED display
- Piezo buzzer
- FTDI USB-to-serial coverter
- TB6612FNG Motor Driver
- 2x 3V geared motors
- 2x wheels with rubber tyres
- 1100 mAH, 3.7V Li-Poly battery pack
- Clear acrylic sheet to cover the PCB

2.2 Component Details

The electric Ray robot is a single PCB-based platform which includes all the components required for a basic robotics project including microcontroller, sensors, motors, motor drivers and power supply.

From the top, an ultrasonic distance sensor module is placed in the front of the robot to measure the distance to the nearest obstacle in the robot's path. This produces a digital echo signal proportional to the distance to the closest object. On the bottom of the board at the same place, three IR sensors are mounted for line detection and following.

The ATmega328 (running at 16 MHz at 5V power) microcontroller onboard contains a Arduino-Uno compatible Optiboot bootlaoder that allows you to upload code through the Arduino IDE. The onboard FTDI USB-to-serial converter IC enables the connection of the microcontroller's serial port to the PC's USB port for programming and serial data transfer. Also connected to this microcontroller are female headers with a standard Arduino Uno footprint to take any Arduino shield compatible with the Arduino Uno.

A 2-color 128x64 pixels 0.96" Organic LED (OLED) display is present to display any messages from the code. This is connected to the ATmega328 through an SPI interface. The Adafruit SSD1306 OLED driver library can be used to control this display. An LED is also available onboard for user control and indication.

The motor driver is based on the TB6612FNG chip which is capable of driving two DC motors at up to 1.2 A continuous current. This driver controls two 3V DC motors with integrated gearboxes for speed control. The motor driver is also capable of driving the motors at adjustable speeds and in both directions.

Power supply is from a 3.7V, 1100 mAH Lithium-Polymer rechargeable battery pack. The 3.7V voltage is stepped up by an on-board boost DC/DC converter to the 5V required by all the components on the board. The battery can be recharged by connecting the robot to a PC through USB. Charging will continue even when

the power switch is in the OFF position. Battery charging will be shut off once the battery has been completely charged.

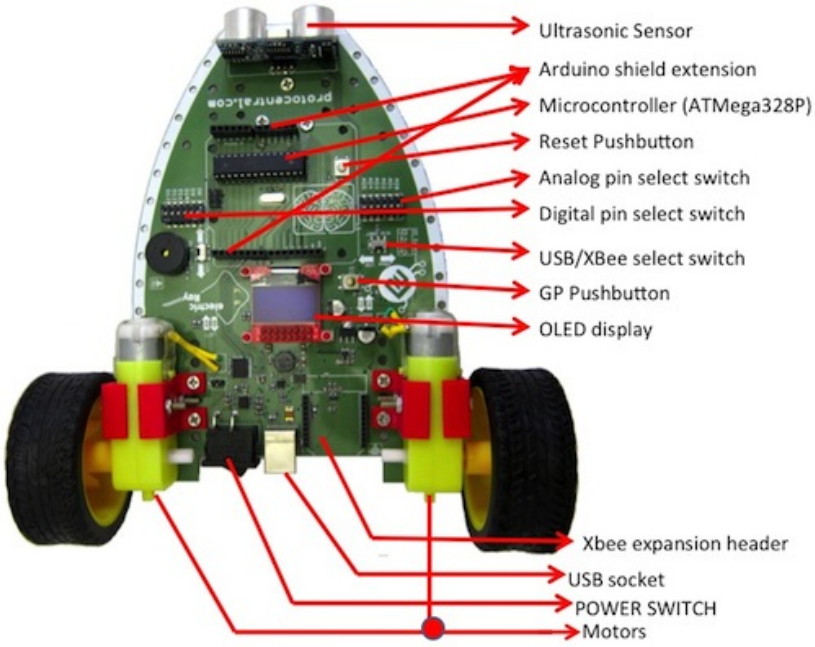

Figure 2.2: Electric Ray Robot

2.3 Programming the robot

To program the robot, connect the robot to your computer via USB. Open the Arduino IDE, and load the sketch located in Appendix B.

You need to tell the IDE which Arduino board you are targeting with your software, so open the Tools - Board menu and choose Arduino Uno.

The Arduino IDE must know which of your USB ports the robot is connected to. The Tools - Serial Port menu lists the available ports. If only one item is shown, click on that one. If two or more are shown, you can disconnect the robot and re-open

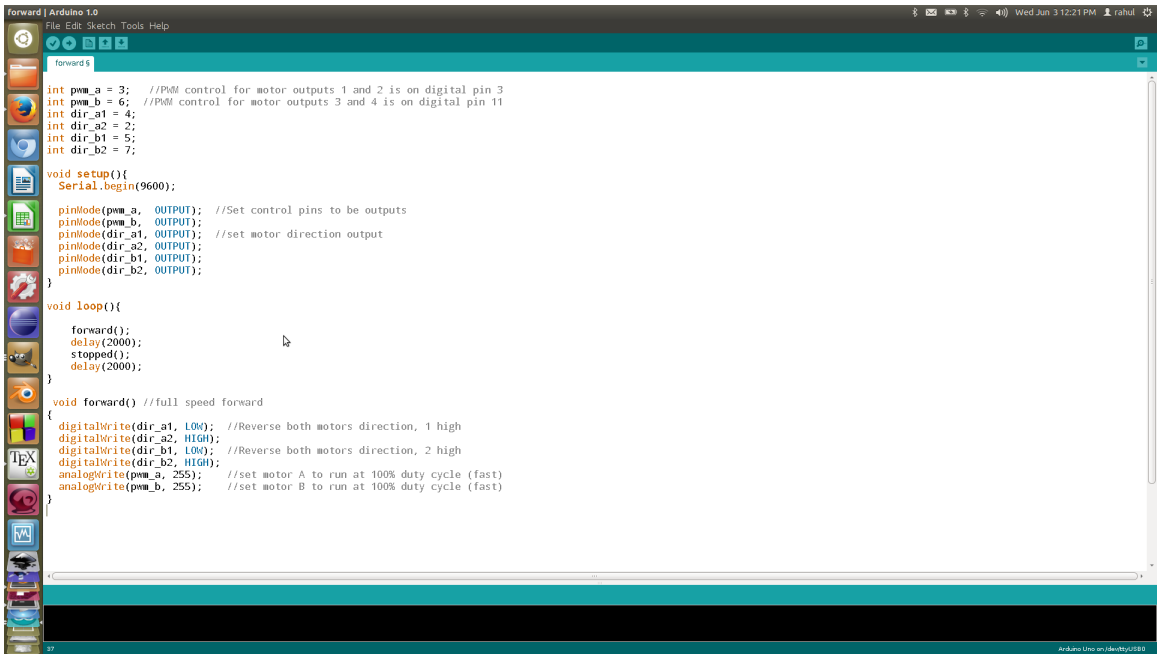

Figure 2.3: Arduino IDE

the menu; the entry that disappears should be the robot. Reconnect the board and select that serial port.

Click the "Upload" button in the top left of the IDE window. Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar of the software. Once this appears, you can disconnect the robot from the USB cable

With batteries in the robot, turn on the power switch and put it on the ground. The robot should start moving forward.

Chapter 3

MiniBloq

This chapter disucusses the design and enhancement of the visual programming environment MiniBloq for robotic platform.

3.1 About MiniBloq

MiniBloq is a graphical development environment for Arduino and other platforms. Its main objective is to help in teaching programming using a drag and drop blocks called Action blocks. It is specially used at elementary, middle and high schools. The MiniBloq version v0.82 used in this research comes in Windows version.

3.2 Enhancement for robotic platform

Minibloq is a graphical code generator with IDE capabilities. It's self-contained and every distribution includes the complete tools needed to compile (or interpret, depending on the selected target) and deploy the code to the selected hardware target.

Every code block is configured in XML. To solve a given problem, the blocks are arranged in a logical manner and then the compiled executable file is transferred to the hardware via data cable. An advantage of Minibloq compared to other graphical programming softwares is that it can be enhanced to add new blocks with functionality specific to your hardware.

MiniBloq is specifically enhanced for Electric Ray robot by adding new blocks by abstracting the pin level details of the hardware. Thus, various blocks for for controlling movement, sensors, display and buzzer of Electric Ray Robot are available to program. The new blocks developed is shown in the Figure 3.1. The enhanced interface of MiniBloq is shown in Figure 3.2.

Figure 3.1: New blocks developed for Electric Ray

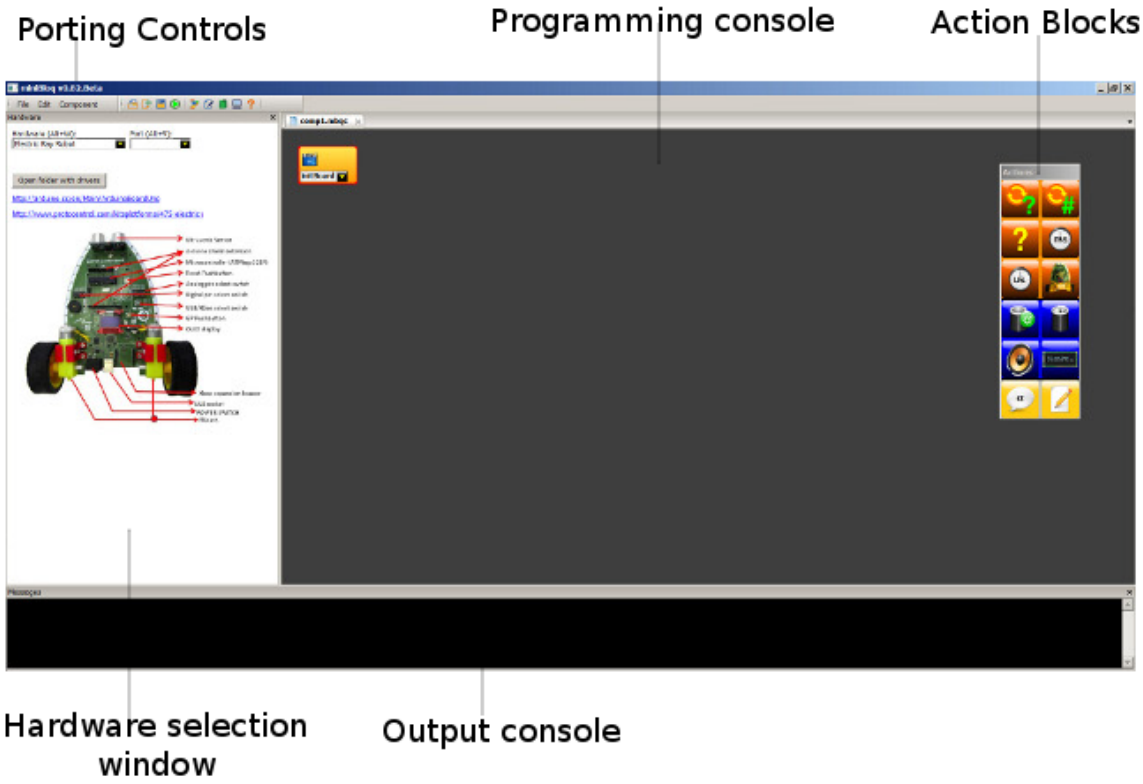

Figure 3.2: Enhanced Minbloq programming environment for Electric Ray

The blocks are designed to be intuitive. A "Hello World" program in the developed graphical tool is written by dragging and droping a block for OLED display, followed by text in the form of string literal as shown in the Figure 3.3. The program when ported to the Electric Ray robot will display the string "Hello World" in OLED.

Figure 3.3: Hello World program in Minibloq

Internally the programming environment converts building blocks to its corre-

sponding C++ code compatible with Arduino. Students interested in modifying the existing property of the building block are allowed to edit the generated C code. For example a code generated for forward movement in the form of block is shown in the Figure 3.4.

```
void forward(int speed) {
    DigitalWrite(D4, false);
    DigitalWrite(D2, true);
    DigitalWrite(D5, false);
    DigitalWrite(D7, true);
    AnalogWrite(PWM3, speed);
    AnalogWrite(PWM6, speed);
}
```

Figure 3.4: Code generated for forward block

Connect the robot using a data cable and choose the connected port in MiniBloq interface. Click on the green play button at the top. This will compile the program and port the program to the robot. A detailed manual on the blocks and is made available at Appendix A

Chapter 4

InBlocks

4.1 Google Blockly framework

Blockly is a newer entrant into the world of Visual Programming Languages for the web as it was released in 2012. The core Graphical User interface of this environment is similar to Scratch, but also quite simplified and less cluttered. Users are presented with a Blocks pane in the left hand-side of the screen, where they can choose between different categories of block functions. The center of the screen, and the majority of the interface is devoted to the work area. This is basically a blank canvas where users can drag and drop blocks and connect them to form the structure of their code. In the lower right-hand corner of the screen is the trashbin, where users can drop chunks of blocks that they wish to delete. The upper side offers buttons to run the program.

Google Blockly is programmed entirely in Javascript. This makes integrating it within any webpage a very simple task, and also greatly simplifies any modifications made to the GUI. Furthermore, Blockly has integrated interpreters that translate

the output of the block strucutures to Javascript, Python, or XML. This is a great advantage, since it allows for easy manipulation and adaptation of the user code to the robot. Finally, the default blocks available to the user can be easily customized using Block Factory. For this research, author has enhanced Google Blockly framework with a support for Android mobile devices.

Chapter 5

Augmented Reality

Augmented Reality is artificial computer generated stimuli which is overlaid onto physical world. It differs from Virtual Reality in a way that it does not suppress perception of physical world but mixes physical reality with virtual content.

A general procedure to generate augmented reality content is described below.

1. **Initialize camera** - This is to capture the real world content.

2. **Capture** - The captured real world content is processed to identify pre-defined patterns.

3. **Identify Marker** - Marker is a predefined pattern available within applications database. Image Processing techniques are used to recognize the presence of marker in the captured video feed.

4. **Retrieving Position/Orientation** - Once the marker is recognized, the position and orientation of the marker is identified and given to the application.

5. **Render Object** - The application renders the virtual content using the position and orientation information got from the above step.

6. **Augment** - In this step the virtual content is layered over the real content.

Figure 5.1 illustrates these steps.

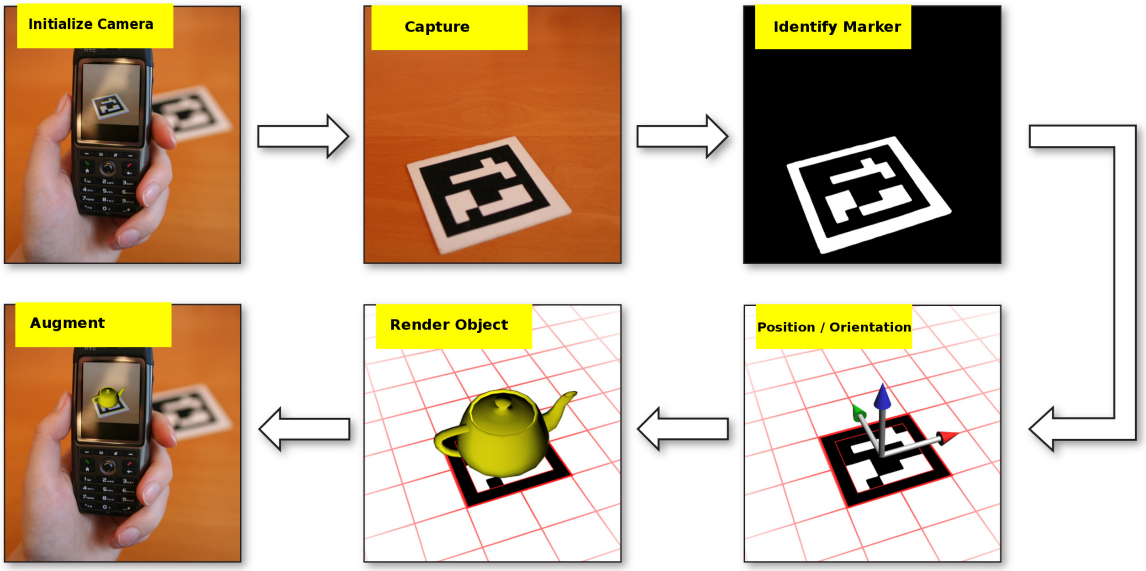

Figure 5.1: Steps to generated AR content

5.1 Vuforia Augmented Reality SDK

Vuforia is an Augmented Reality Software Development Kit (SDK) for mobile devices that enables the creation of Augmented Reality applications. It uses Computer Vision technology to recognize and track planar images (Image Targets) and simple 3D objects, such as boxes, in real-time. This image registration capability enables developers to position and orient virtual objects, such as 3D models and other me-

dia, in relation to real world images when these are viewed through the camera of a mobile device. The virtual object then tracks the position and orientation of the image in real-time so that the viewers perspective on the object corresponds with their perspective on the Image Target, so that it appears that the virtual object is a part of the real world scene.

5.2 Vuforia SDK Configuration

Vuforia SDK can be downloaded from https://developer.vuforia.com/downloads/sdk. To configure the SDK with Android Studio below steps were followed.

- Launch Android Studio
- Select File - Import Project and browse to the root directory of the Vuforia project you want to open
- Proceed in the Import Wizard dialog (click Next, Next) until you reach a page with this message "Alternatively, you can fill in the actual path map in the table below":
- Click to edit
- Enter the actual path to the Vuforia.jar library
- In the Project view, right-click on the Project and expand the view hierarchy so to locate the Vuforia.jar under "app/src/main"
- Right-click on Vuforia.jar to open the context menu

- click on the "Add as library..." option in the context menu
- Alternatively, if you cannot locate the Vuforia.jar in your project hierarchy right-click on the Project and select "Open Module Settings", select "App", then select the "Dependencies" tab and click on the "+" button to Add a File Dependency and browse to the Vuforia.jar file
- Create a folder called "jniLibs" under the "app/src/main" folder under your Android Studio project directory
- Copy the "armeabi-v7a" folder (including the libVuforia.so file located inside it) from the "installdir/build/lib" to the "app/src/main/jniLibs" folder
- the resulting directory structure under your project root should be "/app/src/main/jniLibs/v7a/libVuforia.so"
- Clean and rebuild the project
- Run the app on your device

5.3 Creating the virtual content

The virtual content was created using a 3D modeling tool, Autodesk Maya. The process involved 3 steps -

1. 3D Modeling
2. Texture Mapping
3. Exporting the 3D Model

5.3.1 3D Modeling

3D modeling (or modelling) is the process of developing a mathematical representation of any three-dimensional surface of an object via specialized software. The outcome is called a 3D model. It can be displayed as a two-dimensional image through a process called 3D rendering or it can be used for simulation purposes. The 3D model of the robot created is shown in Figure 5.2

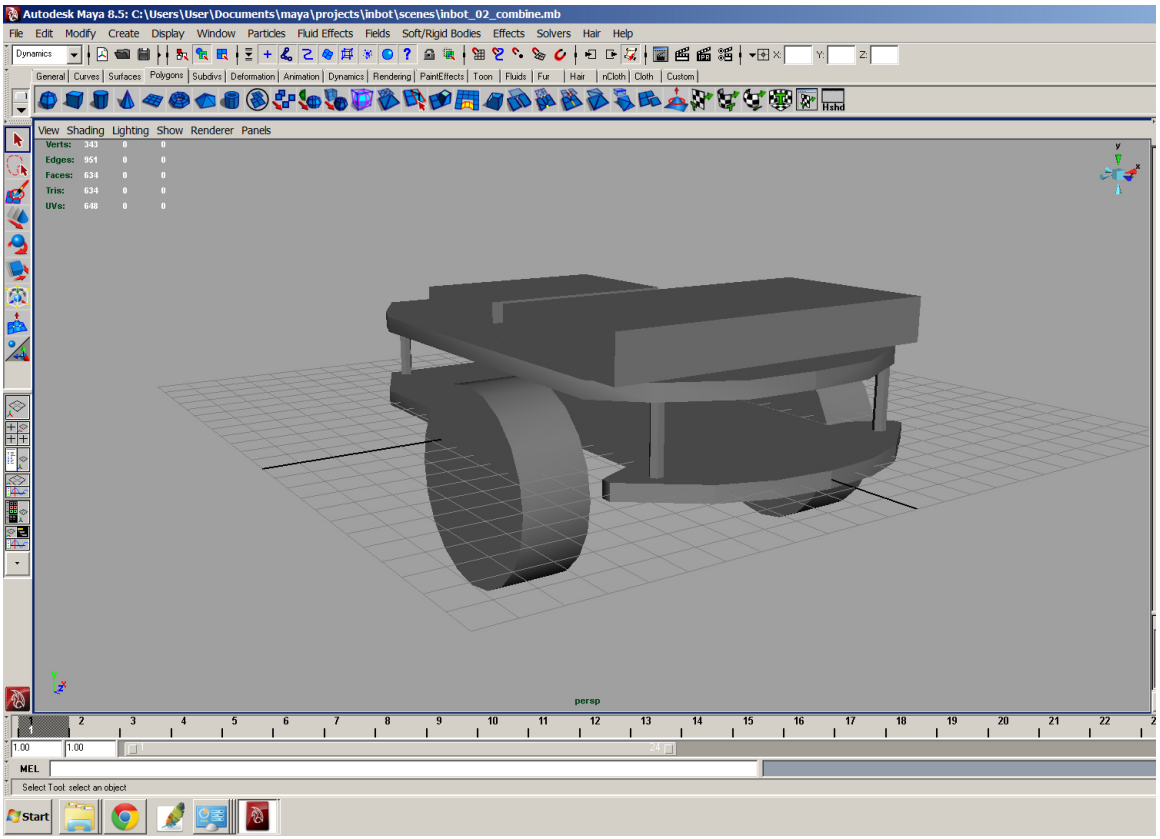

Figure 5.2: 3D model created in Maya

5.3.2 Texture Mapping

Texture mapping is a method for adding detail, surface texture (a bitmap or raster image), or color to a computer-generated graphic or 3D model. A texture map is applied (mapped) to the surface of a shape or polygon. This process is akin to applying patterned paper to a plain white box. Every vertex in a polygon is assigned a texture coordinate which in the 2d case is also known as a UV coordinate. Image sampling locations are then interpolated across the face of a polygon to produce a visual result that seems to have more richness than could otherwise be achieved with a limited number of polygons. The texture map used for the 3D model and the texture mapped 3D model is shown in Figure-5.3 and Figure 5.4 respectively.

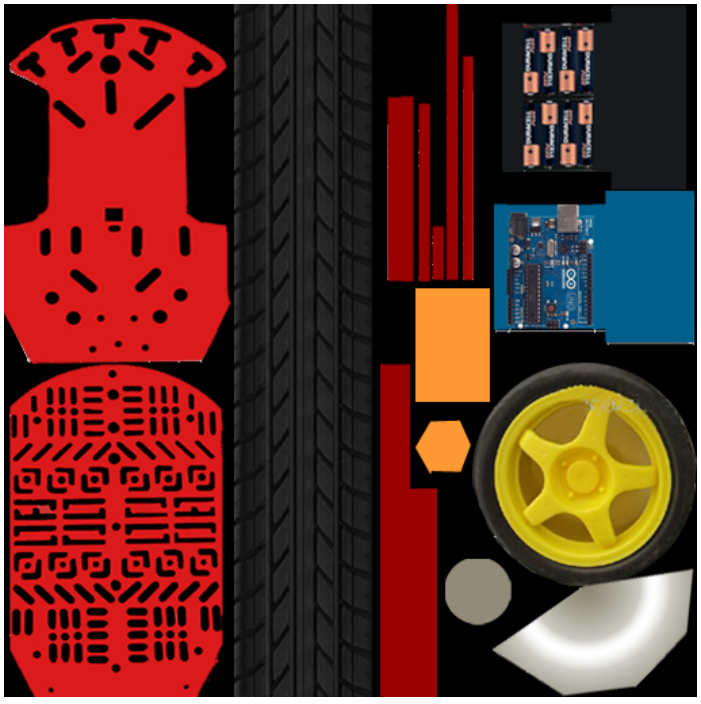

Figure 5.3: Texture Map created using Photoshop

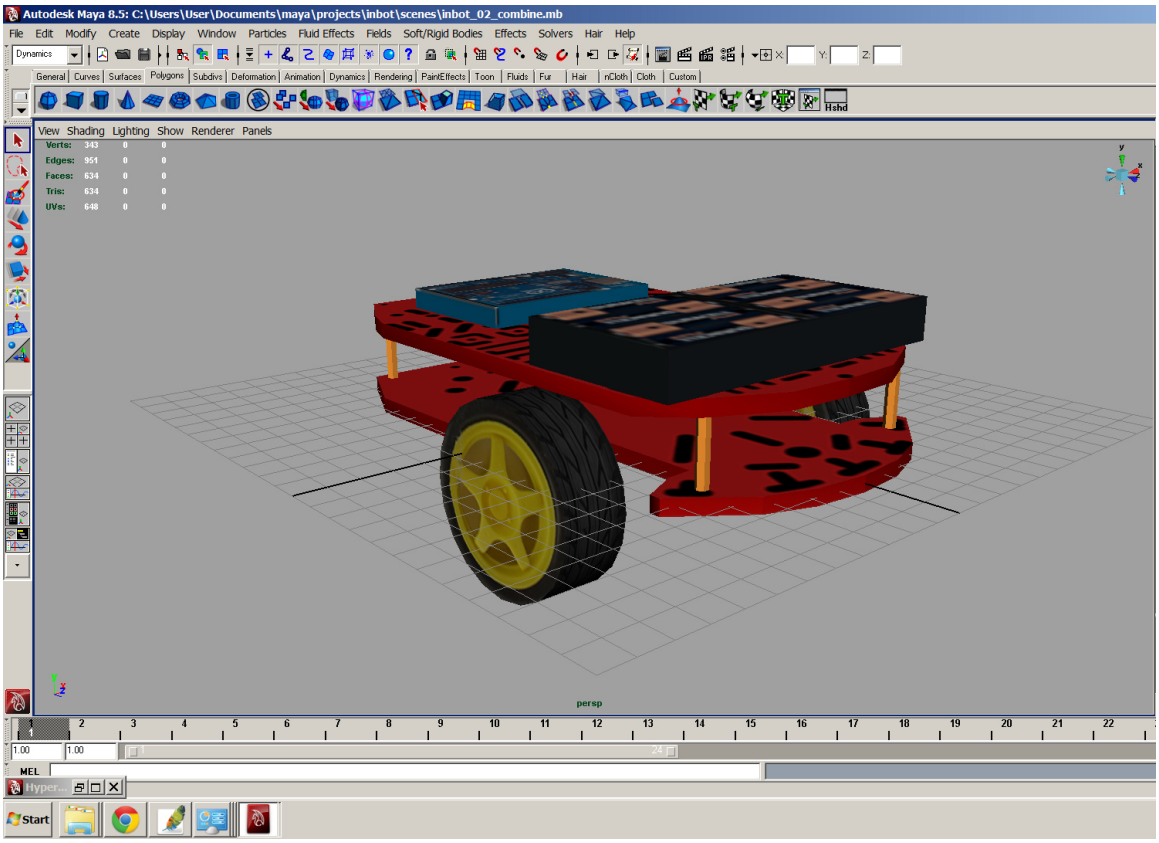

Figure 5.4: Texture Mapped model in Maya

5.3.3 Exporting the 3D Model

Once the model was textured mapped, it was exported to OBJ file format. OBJ file format is a simple data-format that represents 3D geometry using the position of each vertex, the UV position of each texture coordinate vertex, vertex normals, and the faces that make each polygon defined as a list of vertices, and texture vertices. This information was using in Android platform to recreate the virtual content.

Chapter 6

InBot

InBot is a relatively inexpensive differential drive robotic platform capable of teaching basic robotics and sensor integration. It is designed in such a way that the parts can be purchased online and can be easily assembled. A comprehensive Arduino library was written to support the existing peripherals. Once the library is installed, you can program the InBot using InBlocks.

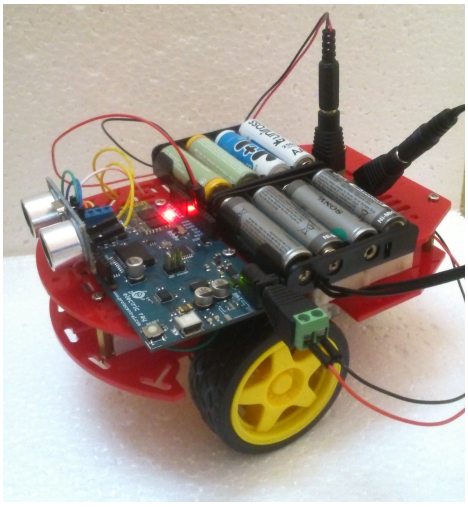

Figure 6.1: InBot

6.1 InBot Hardware

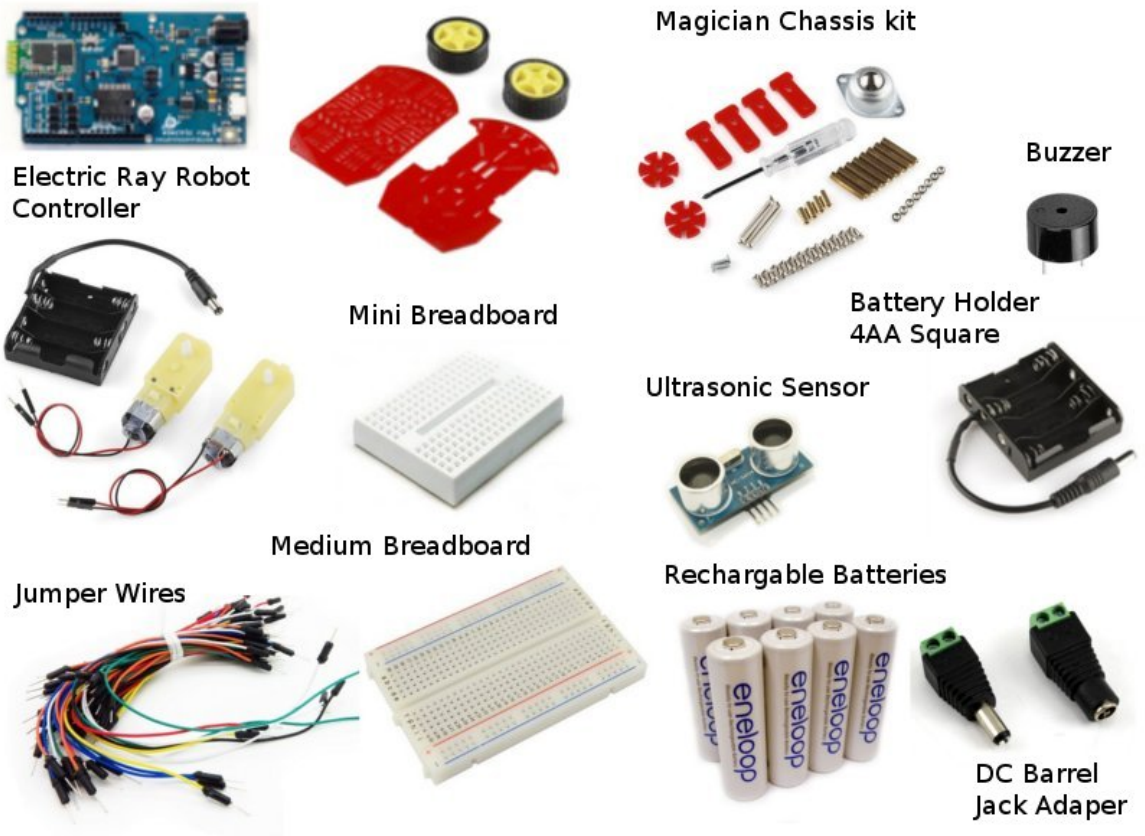

Figure 6.2: InBot hardware components

The Magician Chassis is an economical robot platform which features two gearmotors with 65mm wheels and a caster. Electric Ray is a completely integrated, easy-to-use robot controller fully compatible with the Arduino Uno and programmable with the Arduino environment. With an Arduino-compatible Atmega328P microcontroller, dual motor controllers and a Bluetooth module it can be easily assembled to magician chassis.

Table 6.1: Component Cost and Vendor details

No	Name	Quantity	Price(Rs.)	Purchased from
1	Magician Chassis kit	1	1150.00	Tenet Technetronics
2	Electric Ray Robot Controller	1	3499.00	ProtoCentral
3	Mini Breadboard (Self Adhesive)	1	89.00	ProtoCentral
4	Medium Breadboard (Self Adhesive)	1	159.00	ProtoCentral
5	Ultrasonic Sensor	1	99.00	eBay
6	Buzzer	1	75.00	Tenet Technetronics
7	Battery Holder - 4AA Square	1	165.00	Tenet Technetronics
8	DC Barrel Jack Adaper	2	186.00	Tenet Technetronics
9	Rechargable Batteries	8	849.00	Amazon
10	Jumper Wires (65 piece pack)	1	199.00	ProtoCentral
	Total		6470.00	

6.2 InBot Assembly

A step by step illustration of assembling the InBot is shown starting from Figure 6.4

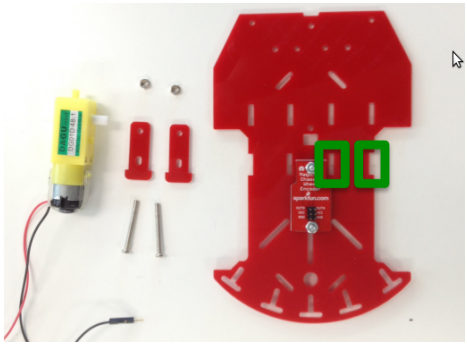

Figure 6.3: Before attaching right motor

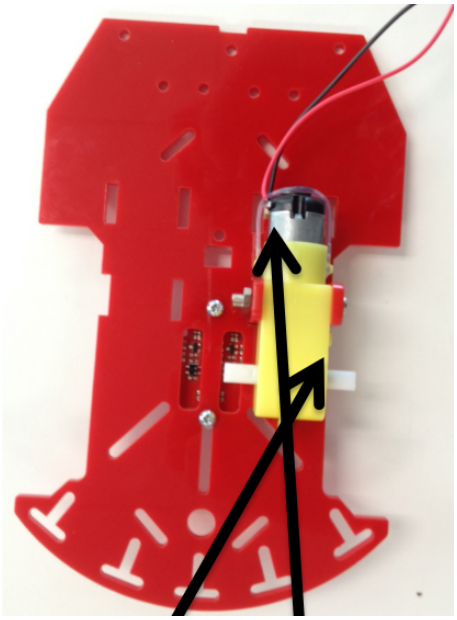

Figure 6.4: After attaching right motor

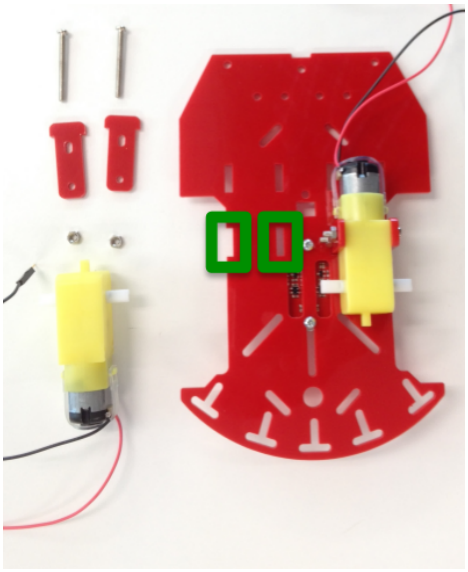

Figure 6.5: Before attaching left motor

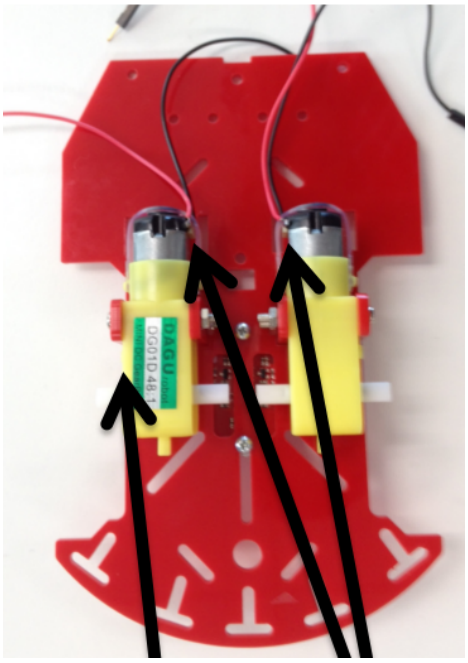

Figure 6.6: After attaching left motor

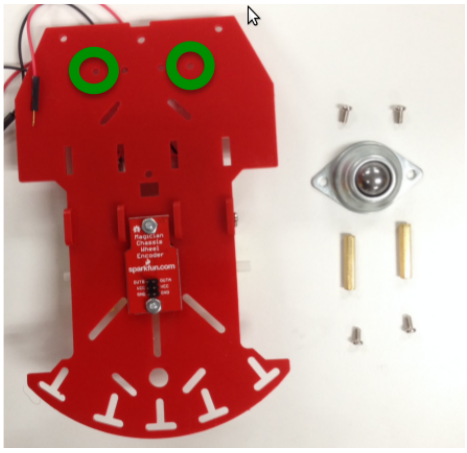

Figure 6.7: Before attaching omni wheels

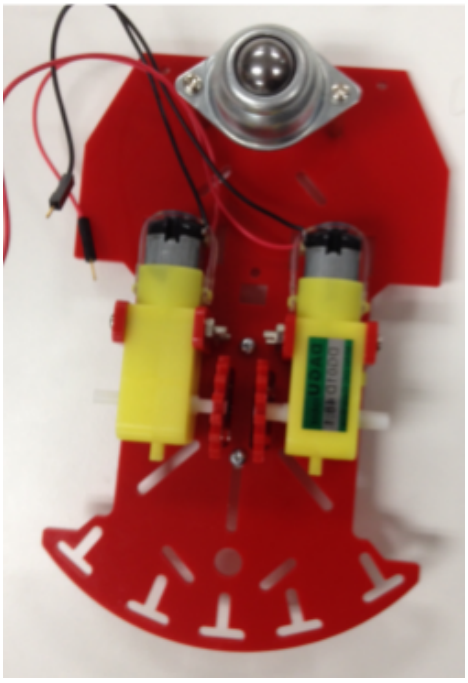

Figure 6.8: After attaching omni wheels

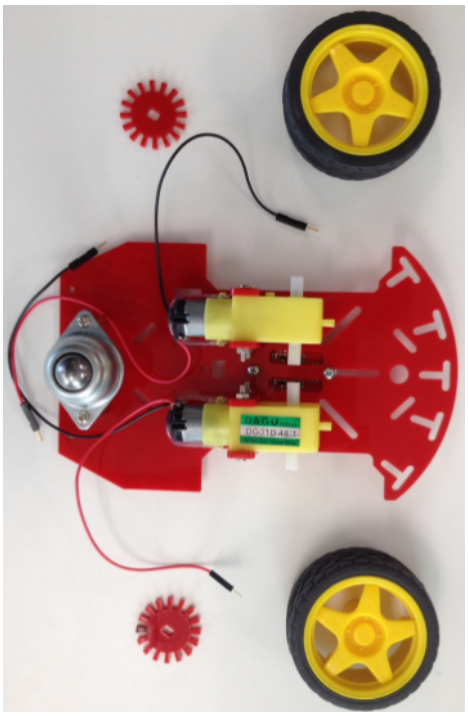

Figure 6.9: Before attaching wheels

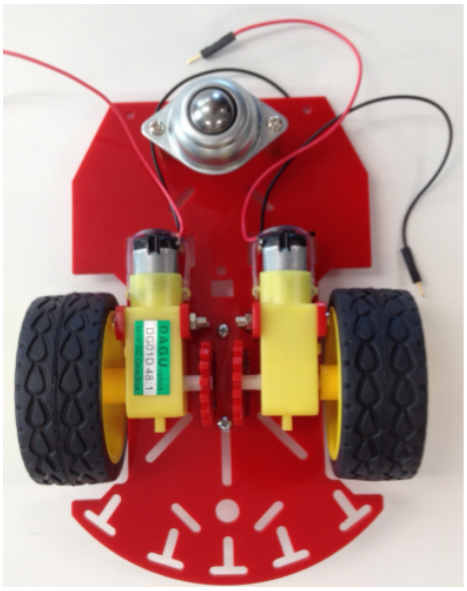

Figure 6.10: After attaching wheels

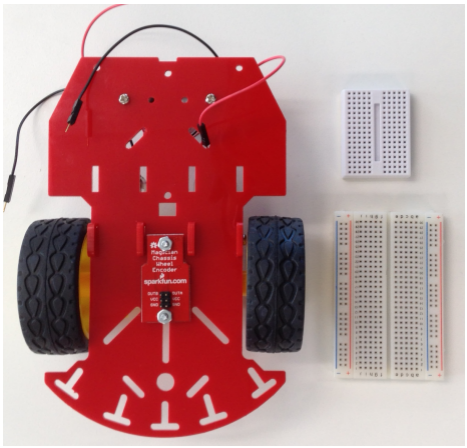

Figure 6.11: Before attaching bread boards

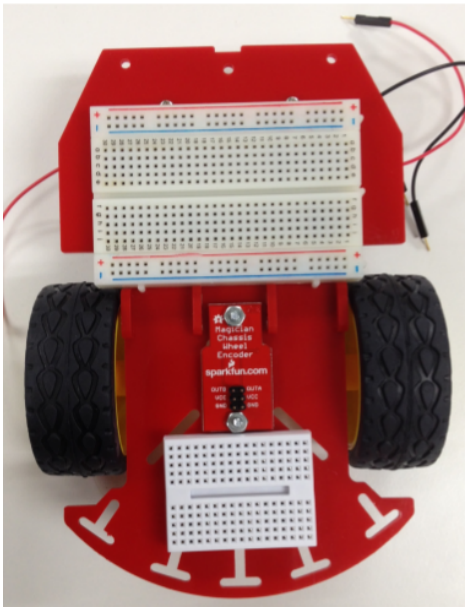

Figure 6.12: After attaching bread boards

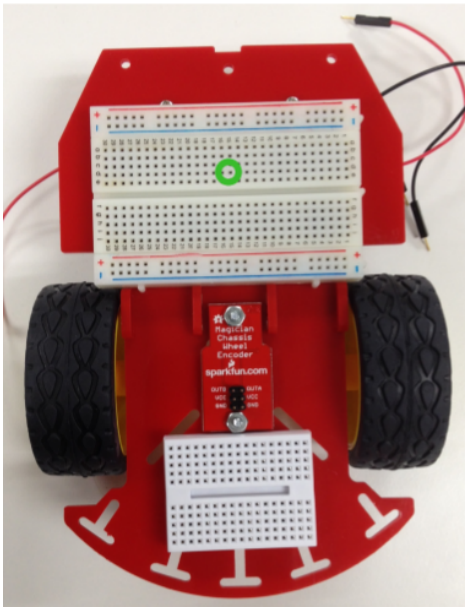

Figure 6.13: Before attaching buzzer

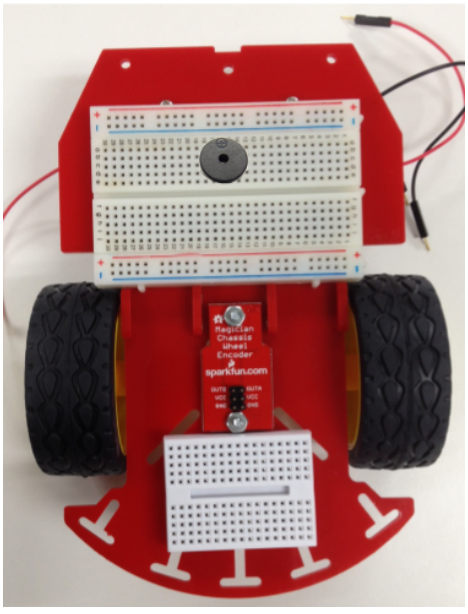

Figure 6.14: After attaching buzzer

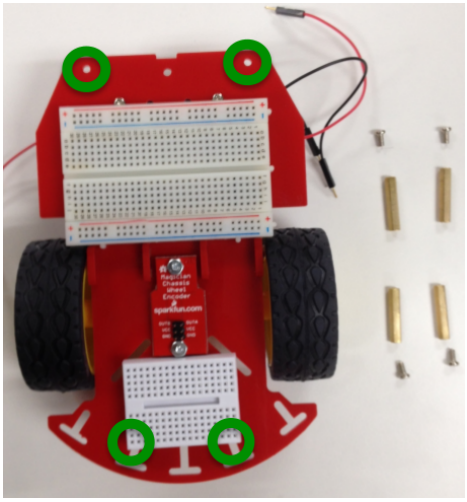

Figure 6.15: Before attaching chassis standoffs

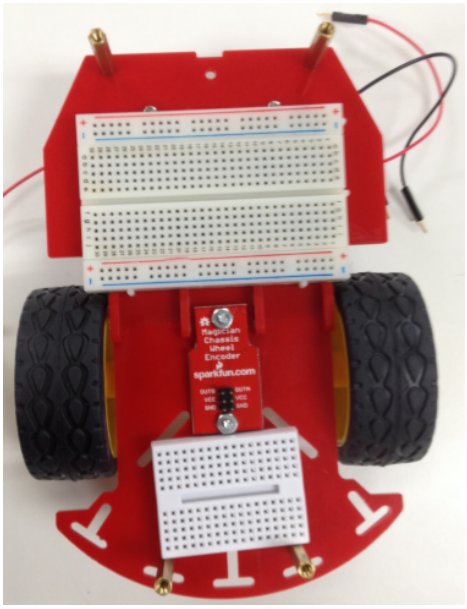

Figure 6.16: After attaching chassis standoffs

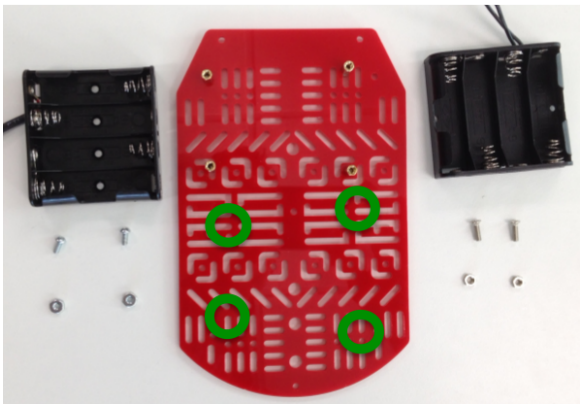

Figure 6.17: Before attaching battery holders

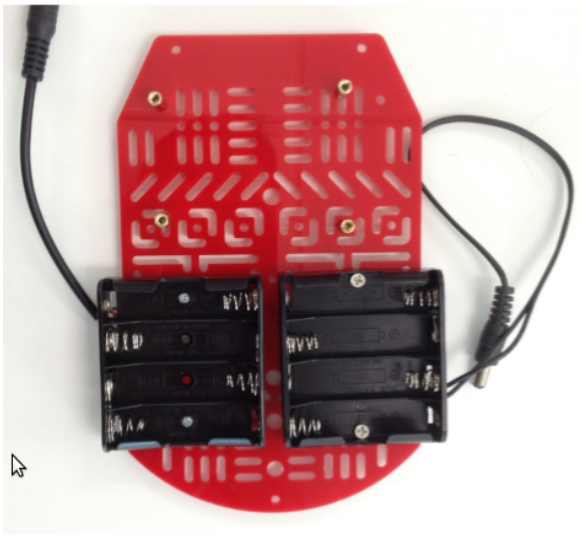

Figure 6.18: After attaching battery holders

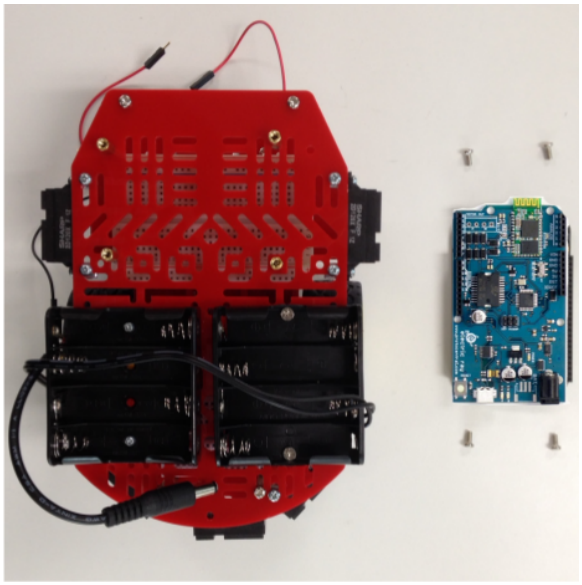

Figure 6.19: Before attaching Electric Ray Robot Controller

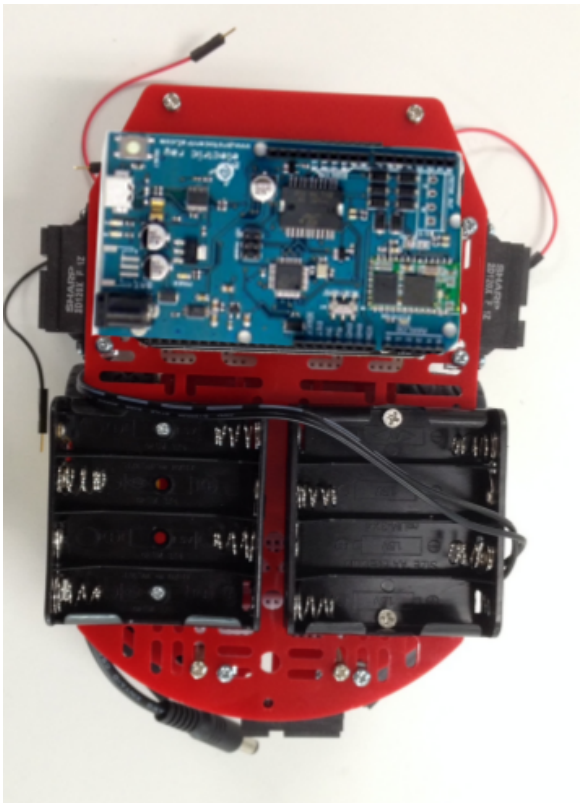

Figure 6.20: After attaching Electric Ray Robot Controller

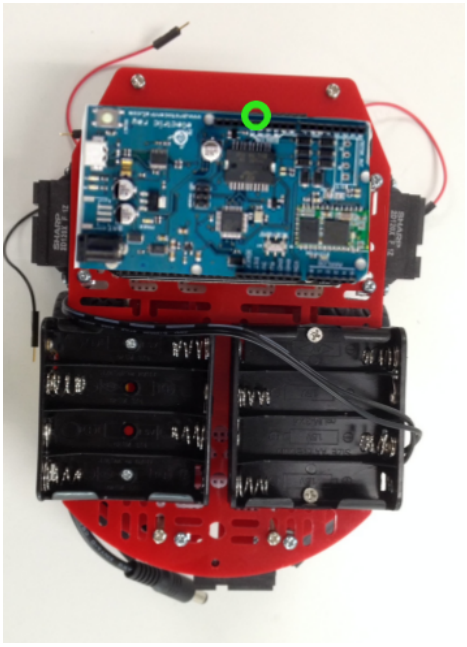

Figure 6.21: Before attaching Ultrasonic sensor

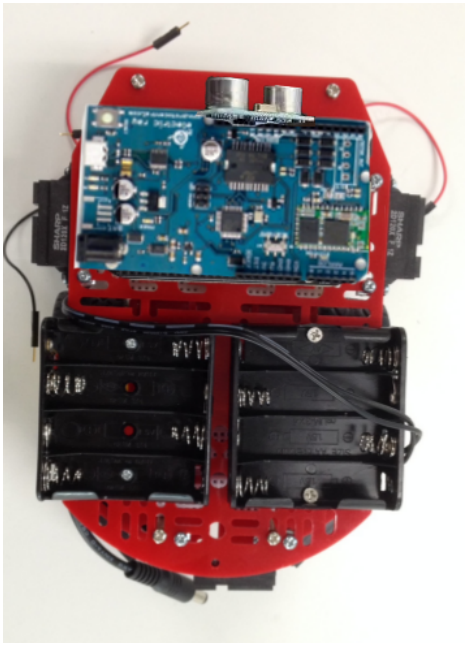

Figure 6.22: After attaching Ultrasonic sensor

Chapter 7

Bluetooth Communication

Chapter 8

Conclusions

Bibliography

[1]

[2] B. S. Fagin, L. D. Merkle, and T. W. Eggers. Teaching computer science with robotics using Ada/Mindstorms 2.0. *ACM SIGAda Ada Letters*, 2001.

[3] L. Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley Publishing Company, second edition, 1994.

[4] F. LastName, F. I. LastName, and F. LastName Jr. Conference paper MUN title. In *Proceedings of the Conference of Sample Conferences*, pages 100–110, Apr. 1996.

[5] F. name Last-name and S. Guy. Journal article SWGC title. *Journal of Sample Journals*, 1(12):1000–1024, 2002.

[6] B. L. Wellman, J. Davis, and M. Anderson. Alice and robotics in introductory cs courses. *ACM New York*, 2009.

Appendix A

MinBloq Programming Manual

A.1 Hello World

Goal: To display "Hello World" on the OLED display of the robot.

Click on the OLED display block. On clicking the red triangle to the right side of the block, a new pop-up expands which shows all the blocks related to display block. Click on the first block which is used to display a string. Click on the red triangle of the string block and select the last block "abc" which will help you type a string. Type "Hello World" as shown below. This completes the program to display "Hello World". Finally port your program on to the robot. The program is shown in Figure A.1

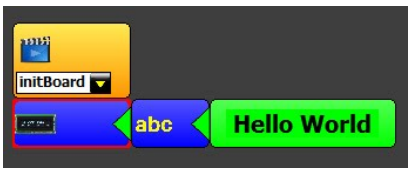

Figure A.1: Program to display Hello World

A.2 Delays

Goal: Blinking "Hello World" display.

Delay block pauses the program for the amount of time specified as parameter. There are two types of delay blocks - the millisecond delay block and the micro second delay block. Click on the millisecond delay block. Then click on the red arrow on the right hand side of the block that just appeared. Now, on the the second row of buttons that just popped up, click on the hash symbol . This is to send a constant value as a parameter. A box with a zero should have attached itself to the right side of the delay box. Type 1000 into it. This will tell the robot to wait one second (1000 ms) before executing any other operations. The below program will blink the "Hello World" display 2 times with a delay of 1 second in between. Complete program is shown below in Figure A.2

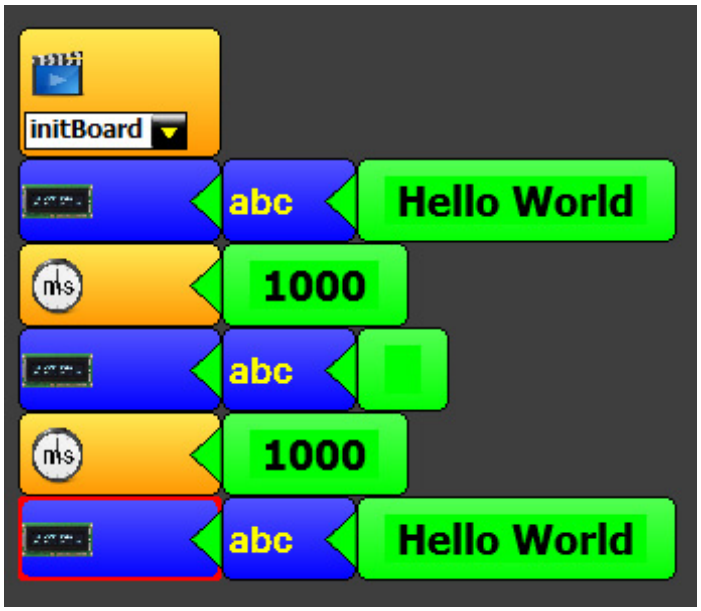

Figure A.2: Program to demonstrate delays

A.3 Variables

Goal: Pass the value for the delay block in previous program as a variable.

Variables are used when we want to store the data in the program. You can use them to hold values and then come back to them when you want to use that value. For example in the previous program, we have used 1000 ms for the delay block at two places. Instead of passing the value directly, we can store the value in a variable and then pass the variable to the delay block.

On the action window in Minibloq, there is a button that looks like a can. An empty can will define a new variable, and can only be used at the top of the program. A full can will reassign a variable's value. Click the empty can to initialize a new variable. Give it a name "delay". Assign a value 1000. Now in the previous program pass the delay variable for the delay block. Complete program is shown below in Figure A.3

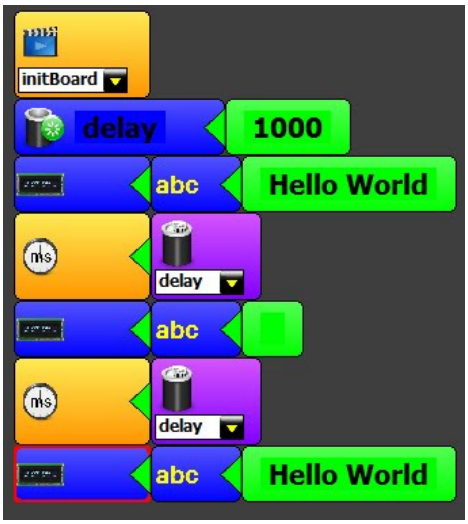

Figure A.3: Program to demonstrate variables

A.4 If Statements

Goal: Display "Hello World" if the value of a variable named "option" is 1 else display "Welcome".

An If statement allows the flow of the program to be changed, which leads to more interesting code.

On the second row of the actions box, there is a "?" block. Click on that to implement an if statement. Three blocks should appear on your screen. The top indicates the start of the if. The second is the else. The bottom indicates the end of the if statement. Clicking on the red arrow on the right side of the top block will open up a boolean menu. If the expression evaluates to true, any blocks between it and the middle "if" block will execute. Otherwise any blocks between the middle and the bottom will execute. Click the equal to sign. This block can accept two values. Select the variable as one value and 1 in the other value as shown below. Complete program is shown below in Figure A.4

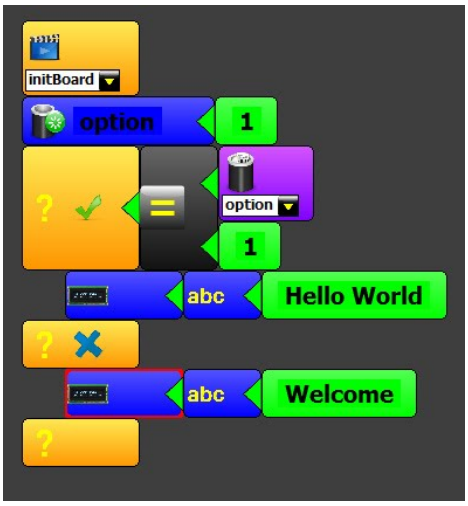

Figure A.4: Program to demonstrate if statement

A.5 While loop

Goal: Blinking "Hello World" continuously.

A while loop is a statement that allows code to be executed repeatedly based on a given boolean condition. The while construct consists of a block of statements and a condition. The condition is evaluated, and if the condition is true, the statements are executed. This repeats until the condition becomes false.

On the actions window, there is a button on the upper left hand corner with a "?" and a cycle on it. This is a while loop block. Click on that to get two blocks on the screen. The top one receives a condition and everything between it and the bottom block executes as long as it is true. Click on the red arrow on the while block. Click on the Tick mark. It means the condition is true. Here we have created a never ending loop. This is necessary as we want the "Hello World" blinking continuously on the display. Complete program is shown below in Figure A.5

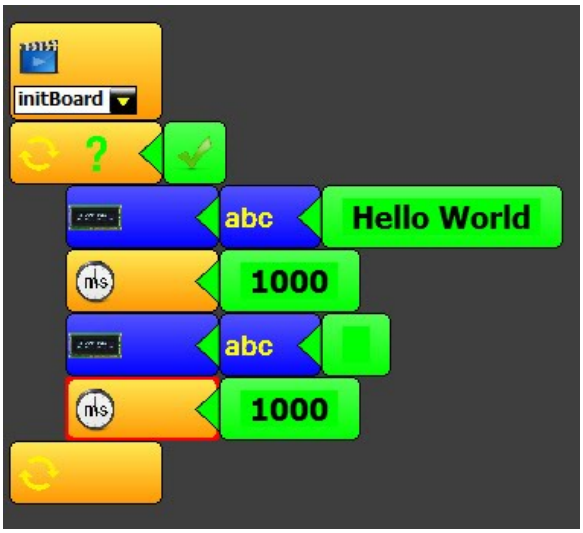

Figure A.5: Program to demonstrate while loop

A.6 Repeat Loop

Goal: To blink "Hello World" 10 times.

A repeat loop is a statement that allows code to be executed for given number of times. The repeat construct consists of a block of statements and a counter value. The counter value is incremented from zero for each block of execution. The statements are executed till the counter value is reached.

On the actions window, there is a button on the upper right hand corner with a hash symbol and a circle on it. This is a repeat block. Click on that to get two blocks on the screen. The top one receives a counter value and everything between it and the bottom block executes till it counter value is reached. Click on the red arrow on the while block. Click on the hash symbol block to give counter a numeric value 10. Add blocks to blink "Hello World" between the two blocks for repeat. The complete program is shown below in Figure A.6

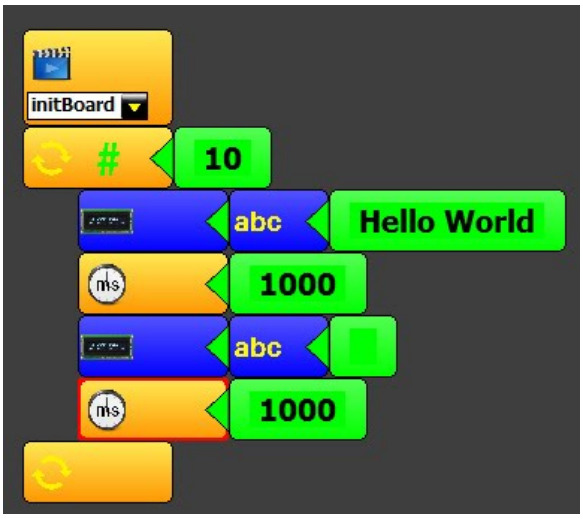

Figure A.6: Program to demonstrate repeat loop

A.7 Movement

Goal: To move the robot forward.

The movement block in the action window helps you to control the movement of the robot. This contains 5 blocks.

1. Forward block
2. Backward block
3. Turn Left block
4. Turn Right block
5. Stop block

To move the robot forward use the forward block. It takes in a parameter speed. Give a constant numeric value between 0 and 255. Add a delay block, to program for how much time the robot should move forward. If we give a value of 250, it means the robot keeps moving forward for 250 ms and then stops. Put the logic in a while loop so that, it keeps on moving forward every 250 ms. Complete program is shown below in Figure A.7

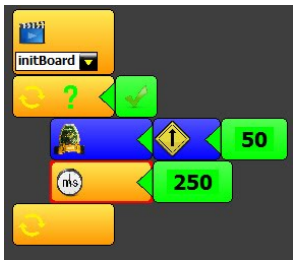

Figure A.7: Program to demonstrate robot movement

A.8 Obstacle Sensor

Goal: To stop the robot when it senses an obstacle ahead.

The obstacle sensor block, positioned in the number window, returns a numeric value (in cm) on how close the robot is to the obstacle. With this value you can set your own threshold, telling the robot on what to do when the threshold is reached. Check if the value returned by the obstacle sensor block is less than 20 cm. If yes, then stop the robot. Else keep moving forward. The complete program is shown below in Figure A.8.

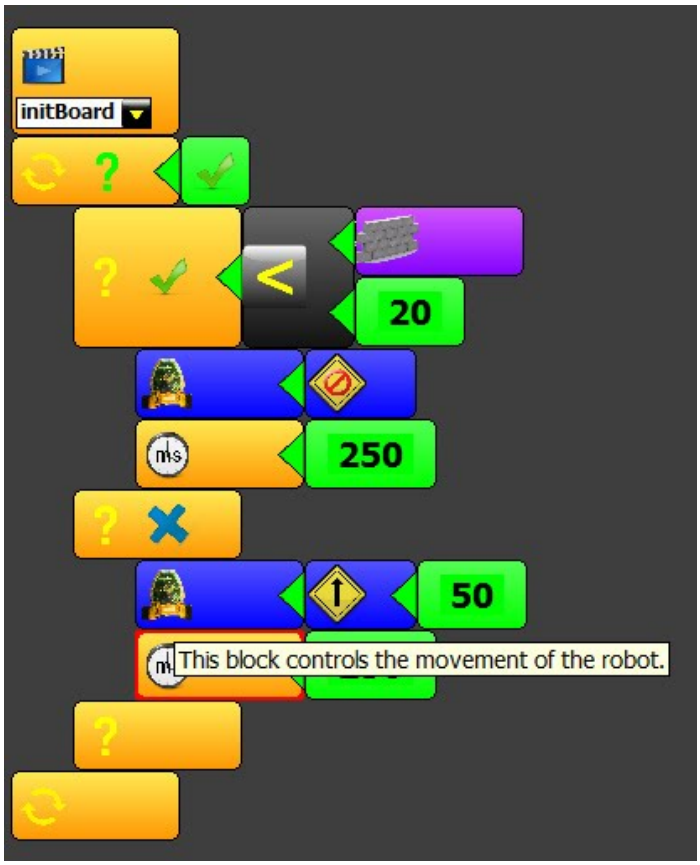

Figure A.8: Program to demonstrate obstacle sensor

A.9 IR Sensor

Goal: Make a floor tile black in color. When the robot enters the black tile, its speed decreases and when it comes out the tile, its speed returns back to normal.

IR sensors use Infra Red (IR) rays to emit and detect the amount of IR light that returns. Its usually used to detect between light and dark surfaces as light colored surfaces reflect more IR light than dark colored surfaces.

The IR sensor block is situated in the number window as it returns a number. Check if each sensor block is returning a value less than 50. That means its passing through a white tile. For white tiles, maintain a speed of 60 for the robot. If the value is greater than 50, its passing through a black tile. For black tiles, reduce the speed of the robot to 30. Complete program is shown below in Figure A.9.

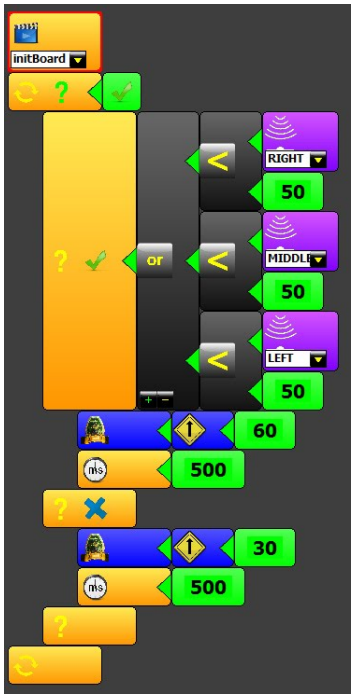

Figure A.9: Program to demonstrate IR sensor

A.10 Buzzer

Goal: To play the notes "c-d-e-f-g-a-b-c".

This lesson will help you use the buzzer to make musical notes. Before you start, make sure that the buzzer is activated with knob for the buzzer is in the right position.

Click on the "Buzzer" block in the action bar. he block takes three parameters -

1. Note - Also referred as the pitch of the sound. This sets the frequency of the sound. By clicking on the red triangle button on the right of the block, user can select from 8 available frequencies (c, d, e, f, g, a, b, C). These are similar to the notes available in music.

2. Beat - This parameter sets the basic unit of time. For example, a note with beat "2" plays for a longer time than a note with a beat "1". Click on the red triangle button to get a number window pop-up. Select hash symbol to give a constant number 1 as input.

3. Tempo - This sets the speed or pace of a given piece of music. So, if its a piece of music, its ideal to have the same tempo for all the notes. For the current task, set the value to a constant number "300".

Repeat the same steps for the notes d-e-f-g-a-b-C. Complete program is shown below in Figure A.10.

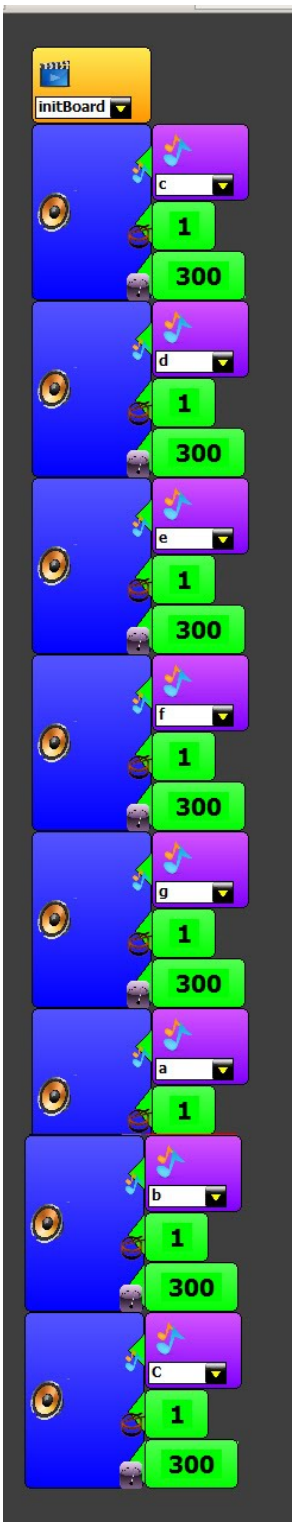

Figure A.10: Program to demonstrate Buzzer

Appendix B

Appendix: How to Add Another One

This is Appendix B.