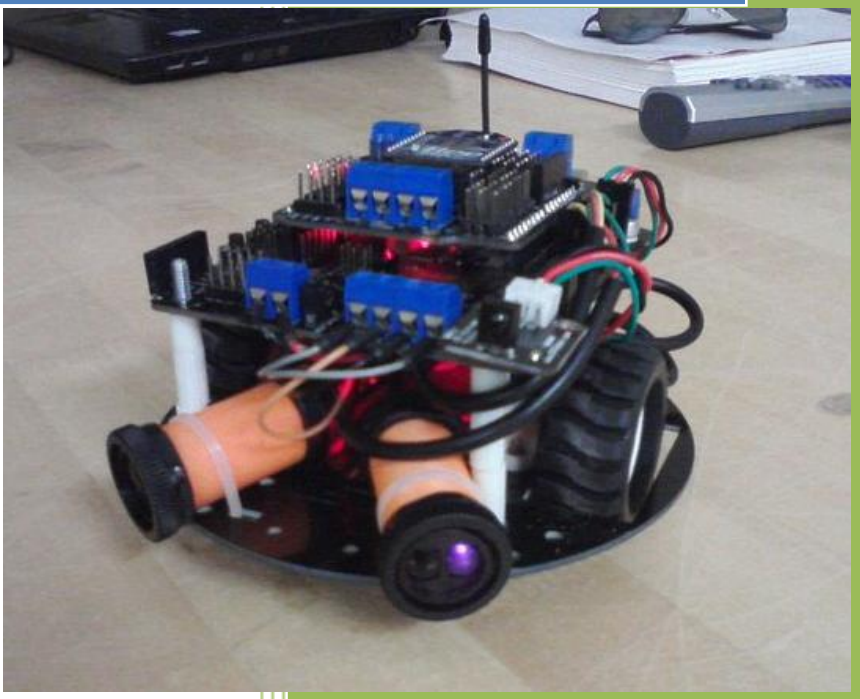
July 25, 2011

An Enhancement to "Co-operating Mobile Robots- 2008"



Report Written By
Chirag Sharma, 100841207
Aerospace Engineering Student, Department of Mechanical and Aerospace Engineering
Carleton University

Report Submitted to
Professor Howard M. Schwartz
Ph.D., (M.I.T.), P.Eng.
Professor, Chair- Department of Systems and Computer Engineering

Report Submitted by
Chirag Sharma, 100841207
Aerospace Engineering Student, Department of Mechanical and Aerospace Engineering

Madeline Harlow, 100817679
Biomedical and Electrical Engineering Student, Department of Systems and Computer Engineering

Introduction to the Document

This document titled "An Enhancement to "Co-operating Mobile Robots- 2008""; dated July 25th, 2011 is written to fulfill the first deliverable requirement of the research conducted in the summer internship at Department of Systems and Computer Engineering, Carleton University.

This research was started and carried out by two first year interns, *Chirag Sharma* and *Madeline Harlow* during two-month tenure. The aim of this research was to make an enhancement model of the robots build by 4th year engineering students as their engineering projects in the academic year of 2008. The research was done under the supervision of *Professor Howard M. Schwartz, Ph.D., (M.I.T.), P.Eng, Professor, Chair- Department of Systems and Computer Engineering* and was sponsored by *Carleton University*.

The document starts off with an introduction on the robotic microcontroller used in this research and then proceeds with different application systems incorporated in the robot model. Each of the application system is well explained in its corresponding chapter. The programming code of each system is also included in the appendices which follow the chapters in this report. Lastly, the report ends with a list of references used in this report and in the research.

This document is the first deliverable of the research. It includes information, details and conclusion carried till the ZigBee wireless network setting. The second deliverable report of this research shall be including information about the Accelerometer sensor and the corresponding research carried using it. The relative positioning mathematical model system shall also be included as well. This second document may be delivered by first year intern *Tara Little*, responsible for carrying out the research for another two-month tenure.

The research interns would sincerely like to acknowledge *Professor John Daly* from *University of Waterloo, Professor Victor Aitken, Carleton University* and *Professor Nagui Mikhail, Carleton University* for constantly helping and providing support to the first year interns during the summer research.

A sincere appreciation would go to *Professor Howard M. Schwartz*, under whose supervision and support were the first year students given such an opportunity and come out to be successful in this research.

Chirag Sharma, 100841207

Madeline Harlow, 100817679

Table of Contents

Chapter 1: DFRduino Romeo Robotic Microcontroller

1.1 Introduction

Romeo originally designed and manufactured by DFRduino is an all-in-one robotic microcontroller specifically designed to carry out easy open-source robotic applications. Supported by thousands of open-source code, the Romeo can be easily expanded with expansion shields for many other purposes [1]. DFRduino provides many advanced versions of the Romeo controller. The version used for this research is 'DFRduio RoMeo V1.0'. Figure 1 shows a glimpse of the controller.

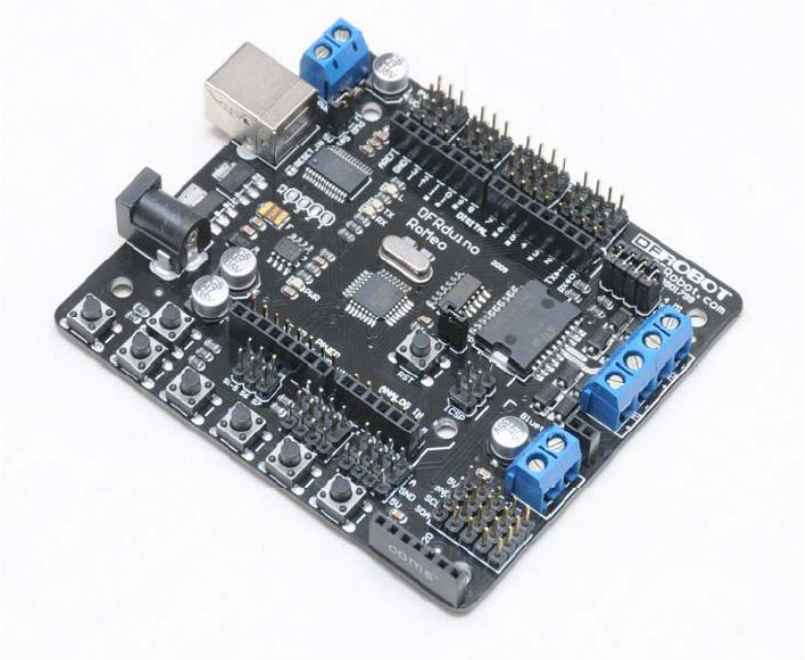


Figure 1: DFRduino Romeo V1.0 Robotic microcontroller [1].

This chapter will discuss the specifications of the controller, and the use of these specifications in the research and setting up of the controller.

1.2 About Arduino Romeo- Specifications

The romeo microcontroller provides a very open-source platform to start an application. The integrated 2 way DC motor driver and wireless socket gives a much easier to set it up for a wireless connection. With its capability to support an expansion shield for a wireless network makes it more of a usable microcontroller.

'The open-source Arduino environment makes it easy to write code and upload it to the i/o board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing, avr-gcc, and other open source software [4].' The software has many versions starting from Arduino 0001 and lasting with the very recent Arduino 0022, the one used in this research. The software uploads the code in the EEPROM library of the board and the board process this program with its Atmel 328 microprocessor.

The Romeo controller comes with an Atmega 328 microprocessor manufactured by Atmel. The high-performance Atmel picoPower 8-bit AVR RISC-based microcontroller combines 16KB ISP flash memory with read-while-write capabilities, 512B EEPROM and 1KB SRAM enough to carry out the research. The processor operates between 1.8 – 5.5 volts and can also be operated by a serial connection supported by the controller board. The Romeo comes with all these features embedded in one board and hence was chosen to be an accurate board. The controller can also be programmed separately by mounting it on the bread board [3].

Next, the board incorporates 14 digital I/O channels out of which 6 are PWM (Pulse Width Modulation) channels. PWM channels are basically made to drive the 2 way motors with a maximum of 2 amperes current and a servo output. The board also has 8 channels for 10 bit I/O analog signals. It supports a USB interface with an ICSP header for a direct program download. The board also integrates sockets for APC220 RF (radio frequency) and DF-Bluetooth module.

The board can also be controlled using 7 integrated key inputs and also supports a display with five I2C interface male header sockets. The board can be powered up by a USB Serial interface and/or External 7V ~ 12 V DC supply and outputs a 5V / 3.3 V DC supply with an external power output for servo motors. DFRduino managed to integrate all these features with desired dimensions of 90 x 80 mm.

Figure 2 shows a detailed schematic diagram of the Romeo microcontroller.

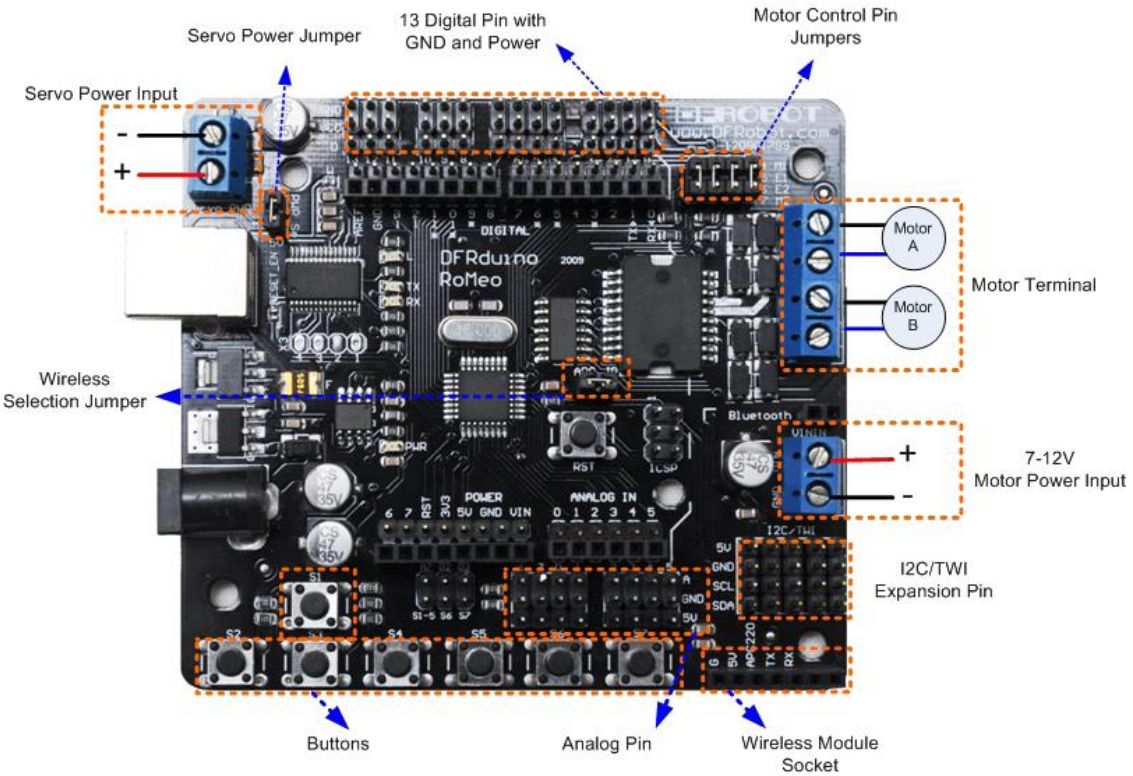


Figure 2: Schematic Diagram of the Arduino Romeo [1].

A major application of this research was to setup a wireless communication network called ZigBee using the expansion board shown in Figure 2. This expansion board supports radio modules called XBee which ended up being the basic connection modules for this network.

The wireless connection was set up by expanding the robotic controller with a 'DFRduino I/O Expansion Shield V5.0' shown in Figure 3.

Figure 3: DFRduino I/O Expansion Shield V5.0 [2].

Using this expansion shield, wireless network called ZigBee was set up which will be discussed in detail later in Chapter 5.

Many programs were written in the Arduino Software (Arduino 0022) to perform many functions. These were to detect the obstacles and move accordingly, controlling the robot through a universal remote, setting up and using the ZigBee network for better communication and using ZigBee for further applications. All these applications are explained later in the report.

1.3 Arduino Software (Arduino 0022) - Programming Software

Arduino provides a very easy open-source platform to program all of its boards (including Arduino Uno and Arduino Dueminilanove). The software has many versions ranging from

Arduino 0001 to Arduino 0022 specifically designed for different processors however; they all are under the same Arduino programming umbrella. The version used for this research is the Arduino 0022.

Before using the software, the drivers for the Romeo board were installed. Arduino makes the board easy-to-handle by installing the driver automatically as and when the board is connected to the host computer using a serial USB connection. A glimpse of the setting up of the arduino programming software is shown in Figure 4.

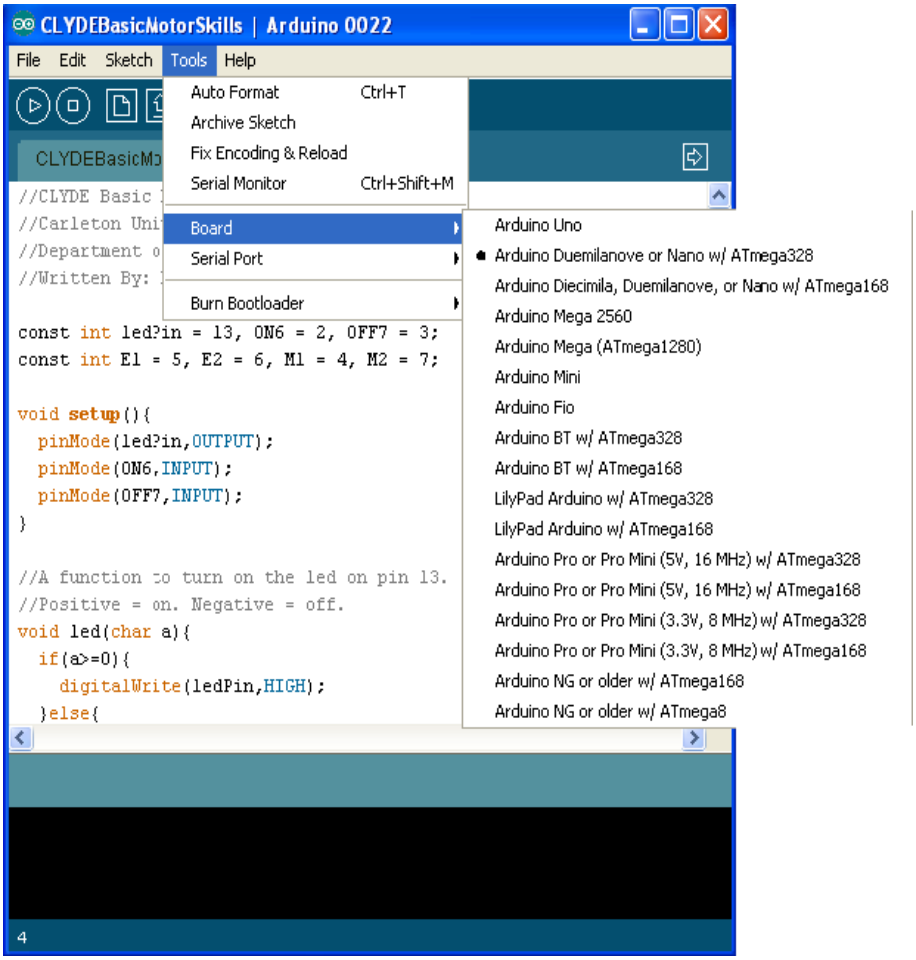


Figure 4: Arduino 0022 programming environment for

Arduino Romeo robotic microcontroller board [Chirag Sharma].

Chapter 2: Motor Control System (MCS)

2.1 Introduction

The Arduino Romeo robotic controller, being a very open source platform provides a separate motor input power supply and 2 way DC motor drive. The separate motor power supply is specifically designed to supply power only to motor controllers, such that power supplied is not compromised with other devices such as the ATMega 328 microprocessor.

2.2 Motor Control System- Pins and Pin Jumpers

Motor controllers on the Romeo board are controlled through analog pins unlike in other boards which have specific designed functions to controls the motors in their software. These pins are analog pins which means that the analog values passed onto them is the frequency of the pulse being sent to the motors. Pins 4,5,6,7 are specifically designed to control 2 motors outputs. 2 pins each- one for the actual speed (frequency) and one for direction (forward or reverse) are allocated to the 2 motors. The maximum frequency that can be passed to the motors is in ratio to the voltage and has the limits of 0 to 200 (200 is the maximum value that can be passed).

'The motor control is implemented by manipulating two digital IO pins and two PWM pins. As illustrated in Figure 4, Pin 4, 7 are motor direction control pins, Pin 5, 6 are motor speed control pins. [1]'.

Pins 4 and 7 (M1 for motor 1 and M2 for motor 2) are digital pins which means that the value passed to them is either 1 (true) or 0 (false). Values used in the Arduino software to control the motors are HIGH (forward) and LOW (backwards).

Pins 5 and 6 (E1 for motor 1 and E2 for motor 2) are PWM (Pulse Width Modulation) pins. The values passed on to them are the frequency in ratio to the input voltage. It has the limits of 0 to 200, where 0 does not move the motor and 200 drives the motor with full voltage.

Combination of pins 4 and 5 (for Motor 1) with pins 5 and 6 (for Motor 2) drives both the motors with desired speed and in desired direction. Arduino makes the Romeo board more reliable by not simply allocating these pins to motors but by using them whenever wanted to. These pins can be used to control the motors only when the motor jumpers are applied. If not, these pins will act as normal analog pins and can be used with analog devices. However, PWM will still be in effect.

The schematic of the motor control pin jumpers and the motor DC outputs can be seen in Figure 5.

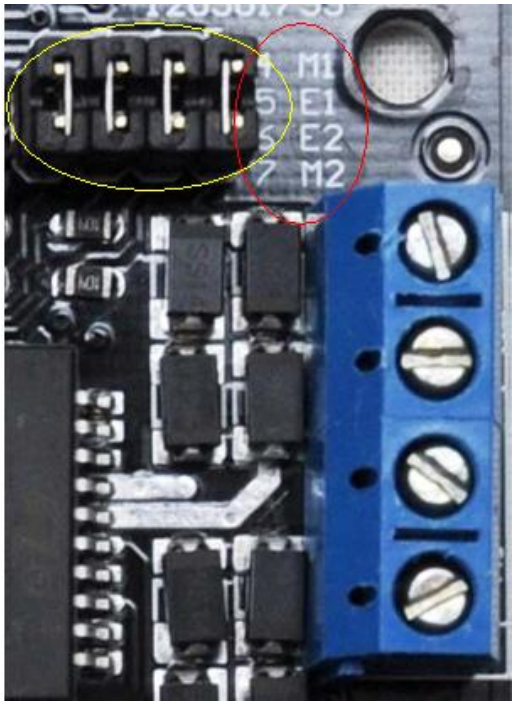


Figure 5: Motor control pin jumpers in yellow and pins in red [Modified from 1].

2.3 Motor Control System- Setting up the Motors

After understanding all the concepts of the motor control system, the motors were attached to the chassis and connected to the board. To control motor 1, pins 4 and 5 were coded in the arduino software and the same was done with pins 6 and 7 to control motor 2. The pin jumpers for pins 4, 5, 6 and 7 were applied in order to allocate these pins to the analog pulse width modulation speed and direction pins.

After the application of pin jumpers, wires were connected to the 2 way DC motor drive outputs (See Figure 4) and then connected to the motors fixed on the chassis. To control the motors, a sample motor control system code was written and uploaded using a serial connection. See Appendix 1 for the sample Arduino code. Upon testing, the motor control system was found to be working perfectly. Figure 6 shows a schematic of the chassis which has the motors connected to the Romeo board.

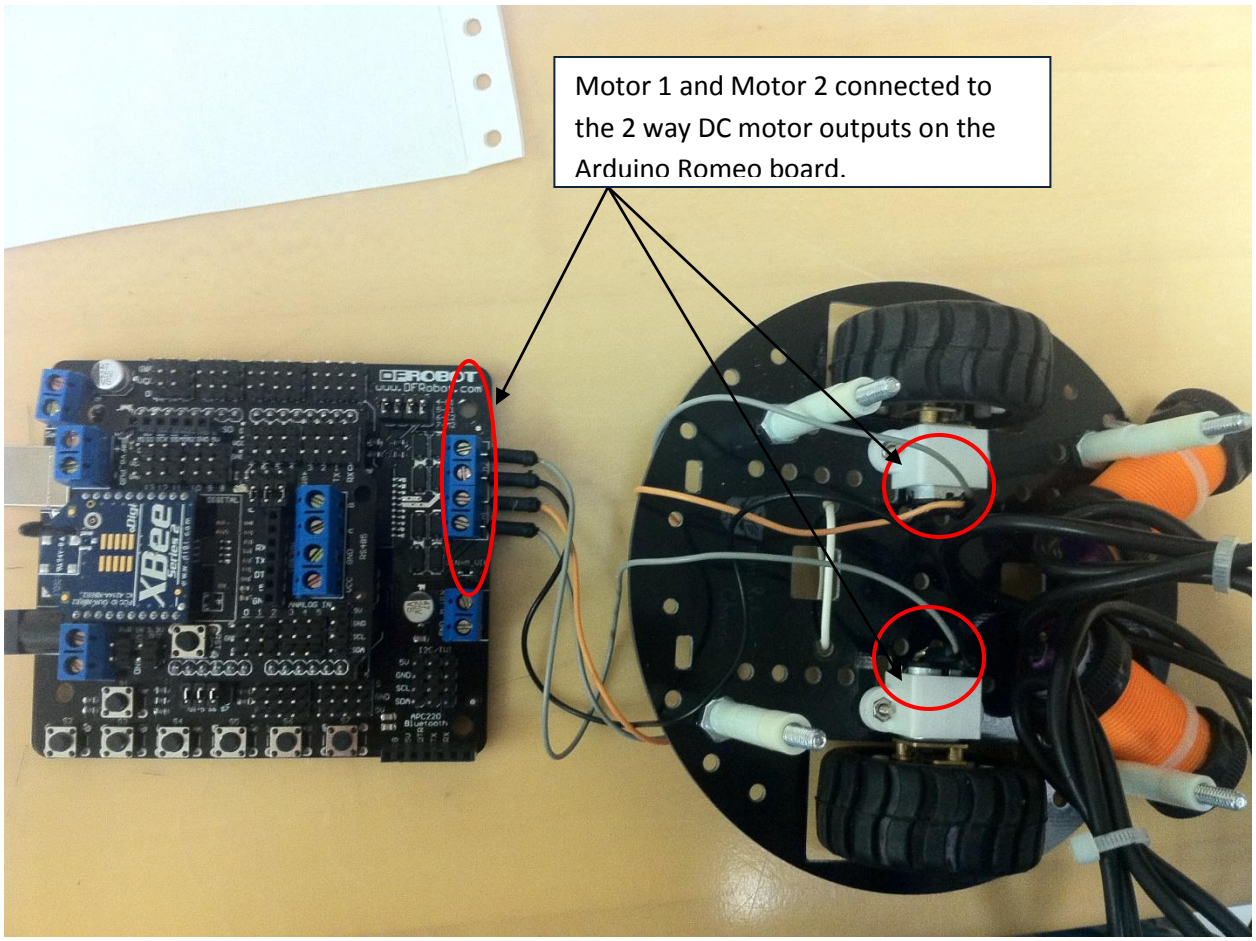


Figure 6: Motor Control connection system on the

Arduino Romeo robotic microcontroller board [Chirag Sharma].

In Figure 5, the gray and the orange cable goes to the motor 2 on the left and the gray and the black cable goes to the motor 1 on the right. The Romeo board was then mounted on the screws with an adjustable height with code uploaded using the serial connection. These motors were controlled using the analog commands **analogWrite** () by passing a value in the range of 0 to 200. This value passed is the frequency in ratio to the input voltage applied.

Chapter 3: Obstacle Collision Avoiding System (OCAS)

3.1 Introduction

OCAS which stands for Obstacle Collision Avoiding System is a self adaptive system application used for this research. It is one of the most advancing system applications to take place in the robotic field of research. The arduino makes it very easy to make this kind of system application on its programming software, upload it on the Romeo board and run on its full functionality.

3.2 Setting up OCAS

The Obstacle Collision Avoiding System (OCAS from here on) has the capability to scan the environment around it and move accordingly when on a robot. Being set on a very basic scale, the OCAS was successfully able to manoeuvre the robot after continuously scanning the environment around it. The open-source quality of the Arduino board again comes in effect making it really easy to program the OCAS on the board.

The system makes use of two infrared sensors capable of detecting obstacles in front of them provided in a given range. The infrared sensors used had adjustable range from 3 cm to 80 cm and gave a digital value. The digital value returned was either HIGH (if no obstacle is detected) or LOW (if an obstacle is detected within range). The infrared sensors used are shown in Figure 7.

Figure 7: Infrared Sensors used for OCAS with an adjustable range [5].

The two infrared sensors used (See Figure 7) provided enough area in front of the robot to detect the obstacle and then steer in the direction opposite to the obstacle. The amount to steer in a particular direction was also determined by the sensor switch. Refer to figure 8 for a complete explanation of the infrared sensor range in front of the robots.

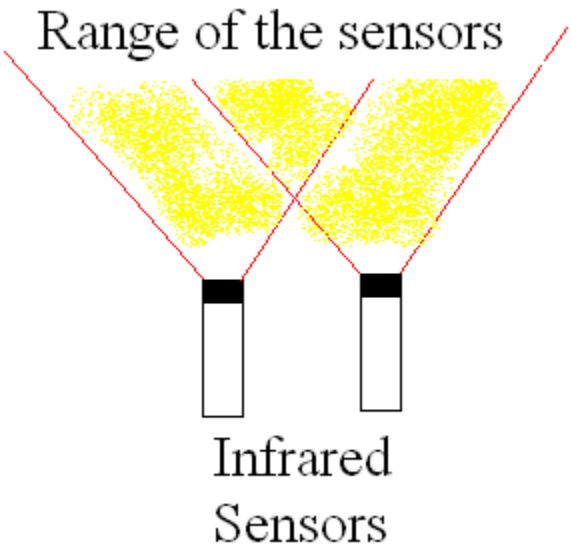


Figure 8: Upfront range of Infrared Sensors [Chirag Sharma].

These two infrared sensors detected obstacles in its range (yellow portion in figure 8), and then processed the code to decide where to move. The code written on the Arduino software checked infinitely for the signals from the two infrared signals and will be discussed in section 3.3. Upon successful uploading of the code onto the Arduino Romeo board, the OCAS worked perfectly fine.

3.3 OCAS Programming Code

The code written to program the arduino board for the OCAS took the advantage of the easy platform supported by Arduino. The two infrared switches were mounted on the chassis with the infrared ports attached to two digital ports on the board. Please refer to Appendix 2 for the OCAS programming code.

The left infrared sensor was attached to digital pin 4 on the board and the right infrared sensor was attached to digital pin 7 on the board. Pins 4 and 7 were chosen because pins 5 and 6 were allocated to the motors through the motor pin jumpers, which controlled the motor direction (High for forward and Low for backward). It should be noted that High and Low are digital values and hence pins 5 and 6 were used by the Motor Control System (Chapter 2).

The code written for the program infinitely checks for the signals received from both infrared sensors while performing different functions or applications such as the Motor control system. If a signal is received, arduino checks the sensor from which it was received (left or right). If a LOW signal is received from the left infrared sensor, the arduino makes the robot veer to the right with an accurate steer and then move continuously and again check for signals. The same process is done with the right infrared signal, but this time the robot moves to the left with an accurate steer and then proceeds continuously. Both the processes are done simultaneously with

the Motor Control System running in the background. These two concurrent systems prove the reliability and the open-source platform system of Arduino.

Chapter 4: Remote Control System

4.1 Introduction

After successful installation of the Motor Control System and the Object Collision Avoidance System, controlling the robot from a remote control handset came up to be a very necessary object of this research. Remote Control System again making use of the easy open-source platform of the Arduino board works simultaneously with the MCS and OCAS running in the back.

4.2 Setting up the Remote Control System

The remote control system gives the operator the advantage of controlling the movement of the robot and deciding its trajectory. The motor control system running in the background makes sure that the voltage (frequency in ratio to the input voltage) is continuously supplied to each of the motors. Its responsibility is also to make sure that the motors stop (sending a null value for the voltage) when given a command from the remote handset. The remote handset used for this system is a Sony Universal Remote, which can work nearly with every infrared receiver.

The OCAS running in the background has the responsibility that even though the operator is controlling the robot, the robot still is scanning its environment and avoiding obstacles in its path.

The technology used in this system makes an efficient use of Infrared light waves. These light waves are transmitted and received by light objects called Infrared Light Emitting Diodes (IR LEDs). These LEDs perform different functions of sending IR light and receiving IR light. An

IR emitter sends IR signals (waves or beats in a more technical way). This IR emitter is always embedded in remotes. These waves are then received by an IR detector. The IR detector again has two different functions.

Some IR detectors give back digital values. These values may be HIGH (1) or LOW (0) and depends if the signals received by the detectors are bounced back upon an encounter with any object. This form is used in the OCAS. Other IR detectors give back analog values. These values are the number of beats that were originally sends by the emitter. This form is used in remote systems. For example, an analog value of 2402 may be send by the IR emitter (when pressing the power button on the Sony remote). Figure 9 illustrates the IR technology.

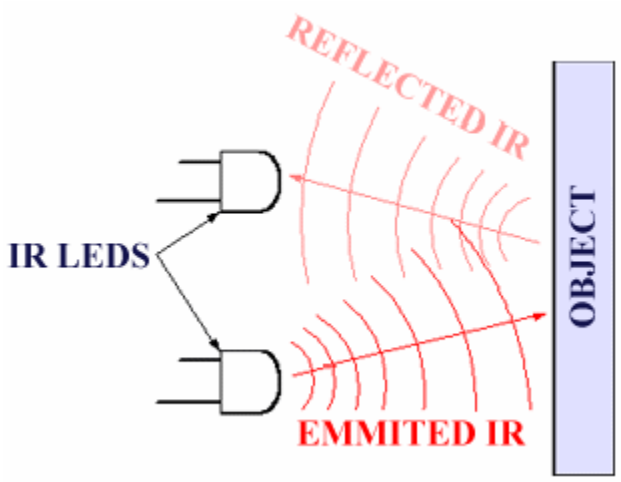


Figure 9: Sending and receiving of Infrared signals [6].

The remote control system was set up using a Sony Universal Remote Transmitter and an IR kit receiver. This universal remote works with any infrared receiver however, a Sony receiver was preferred for a better translation. The IR receiver operates in a 38 KHz frequency and is

generally manufactured by DFRobot, the primary manufacturer for Arduino products. Figure 10 shows the IR receiver used.

Figure 10: Infrared Receiver Used [modified from 7].

4.3 Programming the Remote Control System

The Remote control system was programmed in the Arduino 0022 programming software. The IR receiver shown in figure 10 was mounted on the chassis above the Arduino Romeo board. It can also be mounted just above the Arduino I/O expansion board. This receiver was connected to the digital pin 11 on the board.

Each beat value from the remote control was determined by the program code in Appendix 3. The beat value for each button on the remote was made to display on the serial monitor. These values were then compared in the Arduino memory to take necessary movement routes. Please refer to Appendix 3 for the programming code.

Chapter 5: ZigBee Wireless Network with XBee wireless radio modules.

5.1 About ZigBee network

ZigBee is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power wireless M2M networks. The ZigBee standard operates on the IEEE 802.15.4 physical radio specification and operates in unlicensed bands including 2.4 GHz, 900 MHz and 868 MHz [8].

The 802.15.4 specification upon which the ZigBee stack operates gained ratification by the Institute of Electrical and Electronics Engineers (IEEE) in 2003. The specification is a packet-based radio protocol intended for low-cost, battery-operated devices. The protocol allows devices to communicate in a variety of network topologies and can have battery life lasting several years [8].

5.1.1 The Zigbee Advantage

The ZigBee protocol is designed to communicate data through hostile RF environments that are common in commercial and industrial applications.

ZigBee protocol features include:

- Support for multiple network topologies such as point-to-point, point-to-multipoint and mesh networks

- Low duty cycle – provides long battery life
- Low latency
- Direct Sequence Spread Spectrum (DSSS)
- Up to 65,000 nodes per network
- 128-bit AES encryption for secure data connections
- Collision avoidance, retries and acknowledgements

5.1.2 Mesh Networks

A key component of the ZigBee protocol is the ability to support mesh networking. In a mesh network, nodes are interconnected with other nodes so that multiple pathways connect each node. See figure 11. Connections between nodes are dynamically updated and optimized through sophisticated, built-in mesh routing table [8].

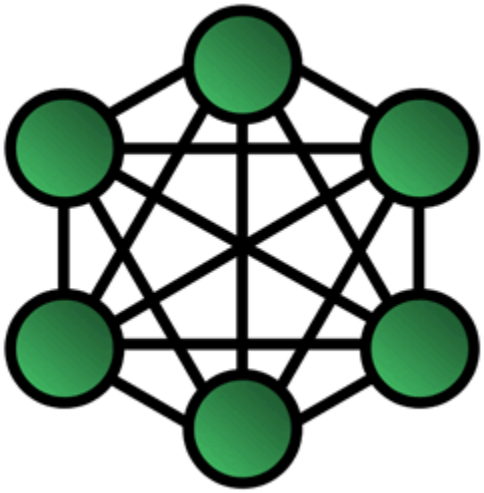


Figure 11: An Example of Mesh Network Topology [8].

Mesh networks are decentralized in nature; each node is capable of self-discovery on the network. Also, as nodes leave the network, the mesh topology allows the nodes to reconfigure routing paths based on the new network structure. The characteristics of mesh topology and ad-hoc routing provide greater stability in changing conditions or failure at single nodes [8].

5.2 About XBee wireless radio modules

XBee and XBee-PRO 802.15.4 OEM RF modules (Figure 12) are embedded solutions providing wireless end-point connectivity to devices. These modules use the IEEE 802.15.4 networking protocol for fast point-to-multipoint or peer-to-peer networking. They are designed for high-throughput applications requiring low latency and predictable communication timing. However, the RF Modules used for this research were XBee series 1.

Figure 12: XBee-PRO 802.15.4 OEM RF module manufactured by Digi International Inc. [9].

Appendix 1: Motor Control System code written on the Arduino 0022 Software

```
//CLYDE Basic Motor Skills
//Carleton University – Engineering
//Department of Systems and Electronics
//Written By: Madeline Harlow 100817679

const int ledPin = 13, ON6 = 2, OFF7 = 3;
const int E1 = 5, E2 = 6, M1 = 4, M2 = 7;

void setup(){
  pinMode(ledPin,OUTPUT);
  pinMode(ON6,INPUT);
  pinMode(OFF7,INPUT);
}

//A function to turn on the led on pin 13.
//Positive = on. Negative = off.
void led(char a){
  if(a>=0){
    digitalWrite(ledPin,HIGH);
  }else{
    digitalWrite(ledPin,LOW);
  }
}

//Stop
void stop(void){
  digitalWrite(E1,LOW);
  digitalWrite(E2,LOW);
}


//Go Forward
void forward(char a, char b){
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}

//Go Backward
void backward(char a, char b){
```

```
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}

//Go Left
void left(char a, char b){
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}

//Go Right
void right(char a, char b){
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}

void loop(){
  //ON button - My Main Function
  if(digitalRead(ON6)==0){
    while(true){
      led(1);
      forward(50,50);
      if(digitalRead(OFF7)==0){
        break;
      }
    }
  }
  led(-1);
}
```

Appendix 2: Obstacle Avoidance Collision System code written on the Arduino 0022 Software

```
const int ledPin = 13, ON6 = 2, OFF7 = 3, IRleft = 4, IRright = 7;
const int E1 = 5, E2 = 6, M1 = 4, M2 = 7;
int check = 0, movingForward = 100, movingLeft = 101, movingRight = 102;

void setup()
{
  Serial.begin(9600); // for serial monitor output
}

//Stop
void stop(void)
{
  digitalWrite(E1,LOW);
  digitalWrite(E2,LOW);
}

//Go Forward
void forward(char a, char b)
{
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}

//Go Backward
void backward(char a, char b)
{
  analogWrite (E1,a);
  digitalWrite(M1,LOW);

  analogWrite (E2,b);
  digitalWrite(M2,LOW);

}

//Go Left
void left(char a, char b){
```

```
  analogWrite (E1,a);
  digitalWrite(M1,LOW);

  analogWrite (E2,b);
  digitalWrite(M2,HIGH);

}

//Go Right
void right(char a, char b){
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}

void loop()
   {
       forward(55,50);
       Serial.print(movingForward);

       Serial.print('\n');

       if(digitalRead(IRleft) == LOW)
         {
           right(50,50);
           Serial.print(movingRight);
           Serial.print('\n');
         }

       if(digitalRead(IRright) == LOW)
         {

           left(50,50);
           Serial.print(movingLeft);
           Serial.print('\n');
         }
      }

   delay(500);
}
```

Appendix 3: Remote Control System code written on the Arduino 0022 Software

```
// example 32.1 - IR receiver code repeater
// http://tronixstuff.com/tutorials > chapter 32
// based on code by Ken Shirriff - http://arcfn.com

#include <IRremote.h> // use the library
int a;
const int ledPin = 13, ON6 = 2, OFF7 = 3, IRleft = 4, IRright = 7;
const int E1 = 5, E2 = 6, M1 = 4, M2 = 7;
int check = 0, movingForward = 100, movingLeft = 101, movingRight = 102;
int receiver = 11; // pin 1 of IR receiver to Arduino digital pin 11

IRrecv irrecv(receiver); // create instance of 'irrecv'
decode_results results;

void setup()
{
  Serial.begin(9600); // for serial monitor output
  irrecv.enableIRIn(); // Start the receiver
}

//Stop
void stop(void)
{
  digitalWrite(E1,LOW);
  digitalWrite(E2,LOW);
}

//Go Forward
void forward(char a, char b)
{
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}
```

```
//Go Backward
void backward(char a, char b)
{
  analogWrite (E1,a);
  digitalWrite(M1,LOW);

  analogWrite (E2,b);
  digitalWrite(M2,LOW);

}

//Go Left
void left(char a, char b){
  analogWrite (E1,a);
  digitalWrite(M1,LOW);

  analogWrite (E2,b);
  digitalWrite(M2,HIGH);

}

//Go Right
void right(char a, char b){
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}

void loop()
{
  if (irrecv.decode(&results)) // have we received an IR signal?
  {

    a = results.value;
    if(a == 2704)  //just waiting for the IR signal to stop.

    //rest is the infraerd sending.
      {
        Serial.println(a);
        stop();
        check = 1;
      }
```

```
//   Serial.println(a); // display it on serial monitor in decimal
   irrecv.resume(); // receive the next value
 }
 // Your loop can do other things while waiting for an IR command
     if(check != 1)

     {

       forward(55,50);
       Serial.print(movingForward); //sending data to other XBee module

       Serial.print('\n');


       if(digitalRead(IRleft) == LOW)
         {
          right(50,50);
          Serial.print(movingRight);
          Serial.print('\n');
         }

       if(digitalRead(IRright) == LOW)
         {

          left(50,50);
          Serial.print(movingLeft);
          Serial.print('\n');
         }
     }

  delay(500);
}
```

Appendix 4: ZigBee Wireless Network with XBee Radio Modules System code(s) written on the Arduino 0022 Software

Code 1: CLYDE_Xbee

```
//CLYDE - XBEE Communication
//Carleton University – Engineering
//Faculty of Computer Systems and Electronics
//Madeline Harlow 100817679
//Summertime Internship
//CLYDE will receive values from JIMMY. If JIMMY detects an
//obstacle CLYDE will react accordingly and turn off his led

const int E1 = 5, E2 = 6, M1 = 4, M2 = 7, ledPin = 13;
byte last = 0;

#include <Wire.h>

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);



//Stop
void stop(void){
  digitalWrite(E1,LOW);
  digitalWrite(E2,LOW);
}
//Go Forward
void forward(char a, char b){
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}

//Go Backward
void backward(char a, char b){
  analogWrite (E1,a);
```

```
  digitalWrite(M1,LOW);

  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}

//Go Left
void left(char a, char b){
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}


//Go Right

void right(char a, char b){
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);

  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}
void led(char a){

  if(a>=0){

    digitalWrite(ledPin,HIGH);
  }else{

    digitalWrite(ledPin,LOW);
  }
}

//Pitcher
//void setup() {
//Serial.begin(9600);

//}

//void loop() {
//Serial.print('x');
```

```
//delay(500);
//Serial.print('o');
//delay(750);
//}

void translateXbee(byte val){
  switch(val){
    case 10: forward(50,50);led(1);stop();
        lcd.println("JIMMY is true."); break;

    case 101: left(50,50);led(-1);

        lcd.println("JIMMY sees something on the right!"); break;

    case 102: right(50,50);led(-1);

        lcd.println("JIMMY sees something on the left!"); break;

    case 2704: stop();led(-1);

        lcd.println("JIMMY turned off."); break;  //power button hit
    default: stop();led(-1);
        lcd.println("JIMMY doesn't make sense."); break;

  }
}

//Catcher

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);

  lcd.init();

  lcd.backlight();
}

int i = 1;


void loop() {
  if (Serial.available()) {
    byte val = Serial.read(); // this will read from the xbee
```

```
    if(last!=val){
      Serial.print(val);
      translateXbee(val);
      last = val;

      lcd.println("\n");
    }
  }else{
    if(i==1){
      lcd.println("No ZigBee.");
      Serial.print("No ZigBee\n");
      i = 2;
    }
  }
}
```

Code 2: XBee_Accelerometer

```
//Pitcher
//void setup() {
//Serial.begin(9600);
//}
//
//void loop() {
//Serial.print('x');
//delay(500);
//Serial.print('o');
//delay(750);
//}

//Catcher

void setup() {
Serial.begin(9600);
}

void loop() {
  if (Serial.available()) {
    for(int i=0; i<3; i++){
      byte val = Serial.read(); // this will read from the xbee
      Serial.print(val);
```

```
      Serial.print('\t');
    }
    Serial.print("\n");
  }
}
```

List of References

[1] DFRobot, *DFRduino RoMeo Users Manual, Version 1.0*, DFRobot, 2011, pp 3, ONLINE, Available: http://www.robotshop.com/content/PDF/dfrobot-romeo-microcontroller-user-manual.pdf

[2] D-Robotics Online, *Arduino IO Expansion Shield (V5)*, 2009, ONLINE, Available: http://www.droboticsonline.com/index.php/arduino-io-expansion-shield.html

[3] Atmel, *ATMega 328P*, Atmel Corporation, 2011, ONLINE, Available: http://www.atmel.com/dyn/products/product_card.asp?part_id=4198

[4] Arduino, *Download the Arduino Software*, Arduino 2011, ONLINE, Available: http://www.arduino.cc/en/Main/Software

[5] DFRobot Drive the Future, *Adjustable Infrared Sensor Switch*, 2011 DFRobot, ONLINE, Available: http://www.dfrobot.com/index.php?route=product/product&path=36&product_id=114

[6] I Kamal, Object Detection using IR Light, *Infrared Proximity Sensor (I)*, IKA Logic Electronic Solutions, 2011, ONLINE, Available: http://ikalogic.com/ir_prox_sensors.php

[7] DFRobot, *Digital IR Receiver Module (Arduino Compatible)*, DFRobot, 2011, ONLINE, Available: http://www.dfrobot.com/index.php?route=product/product&product_id=351

[8] DIGI, Making Wireless M2M Easy, *Zigbee: Low cost, low power, wireless networking for device monitoring and control*, Digi International Inc. 2011, ONLINE, Available: http://www.digi.com/technology/rf-articles/wireless-zigbee.jsp

[9] DIGI, Making Wireless M2M Easy, *XBee-PRO 802.15.4 OEM RF Modules*, Digi International Inc. 2011, ONLINE, Available: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module.jsp#overview