Design and Implementation of manual mode of a Customizable Robot

A Project Submitted
By

K.M.ABDULLAH AL SAFY **ID: 2009-3-55-006**

Under the Supervision of

Dr. M. Mofazzal Hossain
Professor
Dept. of Electronics and Communications Engineering
East West University

&

Dr. Md. Habibur Rahman
Professor
APECE,
Dhaka University

Spring Semester 2014
April, 2014

East West University

DECLARATION

This is to certify that this Project is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. Any material reproduced in this project has been properly acknowledged.

Student name & Signature

K.M.Abdullah Al Safy

Safy

APPROVAL

The Project titled as "Design and Implementation of The Manual Mode of a Customizable Robot" has been submitted to the following respected members of the Board of Examiners of the Faculty of Engineering for partial fulfillment of the requirements for the **degree of Bachelor of Electronics & Telecommunications Engineering** by the following students and has been accepted as satisfactory.

K.M.Abdullah Al Safy **ID: 2009-3-55-006**

26/04/14

Supervisor
Dr. M. Mofazzal Hossain
Professor,
Dept. of Electronics and
Communications Engineering
East West University

26.04.14

Supervisor
Dr. Md. Habibur Rahman
Professor
APECE,
Dhaka University

ACKNOWLEDGEMENT

This was our first project of this kind and considering the challenges of this project we had to face a lot of ups and downs during the process. We needed a lot of support to get the wheel running. We would like to express our heartiest gratitude to both of our supervisors Dr. Md. Mofazzal Hossain and Md Habibur Rahman for their continuous support. Both of their guidance and enthusiasm given throughout the progress of this project has made our robot to come real. Apart from being supervisor they were always there to inspire and encourage us in every possible ways with their advice, brilliant insights and humble guidance. Though words come short while expressing the gratitude we feel, yet from the core of our heart, we would like to thank them both with immense respect.

Our appreciation also goes to our family and friends who have been so tolerant and supportive for us all along. Thanks to them all for their encouragement, love, emotional support and most importantly the belief they had in us. All these form factors have led us to accomplish our goal.

Lastly we are much grateful to the Almighty for keeping us fit and giving us strength to be successful.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The main aim of our project is to build a robot with customizable features. Initially we have managed to complete the control system with some very useful features. We have designed two control systems separately that are – the manual control system and the auto control system. The manual control system using RF technology has been developed by another student of our group. The auto control system is really exciting and challenging because it has two sub features – the collision detection and the cliff detection. The system is designed in such a way that, everything would work simultaneously for different situations. So, the auto control part makes sure that when triggered, the robot can detect static and moving objects in front of it and avoid as necessary as well as detect cliffs and do not jump off and die. And when everything was ready we just merged two control systems to make a powerful and stable system. All of the control system has been developed using Arduino Uno R3 microcontroller development board.

Chapter-1

Introduction

1.1 Introduction

Robotics is the branch of technology that deals with the design, construction, operation, and application of robots. A machine capable of carrying out a complex series of actions automatically, esp. one programmable by computers is defined as a robot. And, Obstacle avoidance refers to the ability of a robot to detect obstacles in its way if there are any and thus make its own obstacle free path. And cliff detection means the ability to detect cliffs and decide whether to move forward or not.

1.2 ARDUINO

Arduino is a popular programmable board used to create projects. It consists of a simple hardware platform as well as a free source code editor which has a "one click compile or upload" feature. Arduino offers an open-source electronic prototyping platform that is easy to use and flexible for both the software and hardware. Arduino is able to sense the environment through receiving input from several sensors. It is also able to control its surroundings through controlling motors, lights and other actuators. The Arduino programming language that is based on the wiring and the Arduino development environment that is based on the processing are used to program the microcontroller found on the board. Due to its open-source environment, one is able to easily write and upload codes to the I/O board. It is also worth to note that Arduino can be run on Linux, Mac OSX and Windows as its environment is written in Java.

1.3 History

Arduino was released in 2005 by students from the Interaction Design Institute Ivrea (IDII) as a modest tool for Mac OSX and Windows. Since then, Arduino has been able to initiate an international-do-it yourself revolution at the electronics industry. The open source microcontroller hardware has been designed in a way that it can easily interface with various sensors (registering user inputs) and driving the behaviors and responses of the external components such as speakers, motors, and LED (responding to the user inputs). The most important feature of Arduino is the ease of programmability hence users with little expertise are able to use it. This aspect has made Arduino one of the most popular tools of choice for designers and artists in creating interactive spaces and objects (Arduino Team).

1.4 Development

While discussing the development of Arduino, it is worth introducing a brief history of microcontrollers. A revolutionary leap in the computing industry was seen in the 1960s following the development of solid state computers (including the IBM 1401), that used transistors to process its operations and a magnetic core memory for its storage (instead of

vacuum tubes), and these enabled an increase in the compactness of the computer hardware. In addition, Jack Kilby's invention of integrated circuits in 1959 enabled circuits and transistors to be fused into tiny chips of semiconducting materials (like silicon) as well as further miniaturization of the computer component. The other crucial development made in the same decade was the high level computer programming languages, written in symbolic languages such as plain English, and this made computer codes somehow easy to learn and read than the earlier machine languages that consisted of letters and numbers only. This development enabled individuals with few years of expertise to carry out the basic operations on a computer. FORTRAN (for the scientific calculators) and COBOL (for business application) were the two main languages that were introduced in that period. The microprocessor was one of the greatest innovations in the history of the modern computer in the 1970's. Initially, the microprocessor miniaturized all the hardware components of CPU to fit into one, tiny, integrated circuit, popularly known as the microchip. The microchip became the major driving component of the microcontrollers including Arduino which is made up of a microchip, input/output hardware and memory storage hardware for sensors. The microprocessor, due to the small form factor, was incorporated into a surfeit of electronic devices ranging from personal computers to calculators and are still used up to date. More programming languages were also developed in the 1970s and 80s including C, C++ and Java for applications in science and business. (Massimo, 2005)

1.5 Evolution

Having looked at the evolution of microcontrollers, there have been recent incarnations of the microcontrollers that have been designed in a way to fulfill the needs of hobbyists and casual users who happen to have a limited technical knowledge. In other words, the microcontrollers have moved from the more complex requirements in the scientific, business or commercial fields. Before the invention of Arduino, the PIC microcontroller board that was introduced by general instruments in 1985 was one of the most used tools for the electronic enthusiasts. The reasons as to why the PIC microcontroller board was preferred were the speed and ease of its programming through simple languages including PBASIC. An additional reason was that it was able to store programs on a flash memory chip that enabled the instructions on the board to be reprogrammed or erased at will with an infinite number of possibilities. It also supported output devices such as LEDs and motors as well as input sensors. There are other popular boards for the hobbyists including BASIC Stamp and wiring which are some of the microcontroller boards that were designed for tangible media explorations and electronic art. The two boards share the advantages of ease of rapid prototyping and simplicity of programming.

It was in 2005 when the Arduino team was formed in Italy and it consisted of Barragan Massimo, David Cuartielles, Gianluca Marino, Dave Mellis and Nicholas Zambetti. The main goal of this team was to develop an electronic prototyping platform that would simplify the wiring platform and make it accessible to the non-technical users especially in the creative fields. The Arduino, therefore, incorporated several characteristics including a programming

environment that is based on the processing language that was conceived by Casey Reas and Ben Fry and other artists and designers. Arduino also incorporated the ability to program its board using a standard USB connection with a low price point (Wheat, 2001).

1.6 Past and Present

Within its first 2 years of existence, Arduino achieved rapid success where more than 50, 000 boards were sold. By 2009, Arduino had more than 13 different incarnations with each having a specialized application. For instance, Arduino Mini was a miniature to be used in small interactive objectives, Arduino BT was built with Bluetooth capabilities, and Arduino Lilypad for wearable technologies projects. Today, the Arduino microcontroller is a popular prototyping platform across the world and it is a good example of how software and hardware technologies that were originally created for business, military or scientific applications have been repurposed so as to serve the needs of people developing projects in new media and arts and design. The following list includes the present and past Arduino boards: Arduino shields, Arduino USB, Arduino single-sided serial, Arduino serial, Arduino Mega, Lilypad Arduino, Arduino Fio, Arduino BT, Mini USB adapter and Arduino Mini. In the development of Arduino, the following silverware times have been developed: in 2005, a project was started to develop a device that would control the student-built interactive design projects which was cheaper compared to other prototyping systems that were available at the time. The founders of the project, David Cuartielles and Massimo Banzi, named the project Arduin of Ivrea. They then began producing boards at a small factory in Ivrea in the Northwestern Italy. In September 2006, they released Arduino Mini and later in October 2008, Arduino Duemilanove was developed and was earlier based on Atmel ATmega 168 and later, on ATmega328. Arduino Mega was then released in March 2009 and was based on Atmel ATmega 1280. More than 300,000 units or Arduino in May 2011 were in use across the world. Arduino Leonardo was later released in July 2012 and is based on Atmel SAM3X8E that has an ARM Cortex-M3 core. Arduino Micro was released in November 2012 and is based on Atmel ATMega32u4. Arduino Uno has been named to mark the new Arduino 1.0 where version 1 and Uno will be the reference model for the Arduino platform. For the purpose of comparing with the previous versions, Arduino Uno will be used in the project. Arduino Uno is a microcontroller board which is based on the ATmega328 that has 14 digital I/O pins. Among these pins, six of them may be used as PWM outputs, one as a 16 MHZ crystal oscillator, 6 as analogue input, one as a USB connection, one as an ICSP header, and one as a power jack and reset button. Everything needed to support the microcontroller is contained on the board like connecting to a computer with a USB cable and power it using an AC-to-DC battery or adapter.

Arduino Uno differs from other preceding boards due to its features which include ATmega8U2 that is programmed as a USB-to-serial. Arduino Uno may be powered either through the USB connection or using an external power supply. The selection of power source is automatic and

the power pins include: VIN which is the input voltage on the Arduino board while using the external power source and this is as opposed to the 5 volts from regulated power source or USB connection. Voltage may be supplied through this pin, or while supplying voltage through the power jack, it may be accessed through this pin. The regulated power supply is 5V that is used to power microcontroller as well as other components on the board. It can be supplied by the USB or any other regulated 5V supply. A 3.3V supply is generated by the on-board regulator or it can come from VIN through the on-board regulator. The maximum current drawn is 50 mA. The GND ground pins- the ATmega328- have 32 KB including 0.5 KB that is used for the boot loader. It has 2 KB of the SRAM and a further 1 KB of EEPROM that can be written or read with the EEPRO library. The maximum width and length or the Uno PCB is 5.3 and 6.8cm respectively. The power jack and the USB connector extend beyond these dimensions. There are four screw holes that allow the board to get attached to a case or surface. It should be noted that the distance between the digital pins 7 & 8 is 0.16cm.

1.7 Embedded System (ES)

The microprocessor-based system is built for controlling a function or range of functions and is not designed to be programmed by the end user in the same way a PC is defined as an embed system. An embedded system is designed to perform one particular task albeit with different choices and options.

Embedded systems contain processing cores that are either microcontrollers or digital signal processors. Microcontrollers are generally known as "chip", which may itself be packaged with other microcontrollers in a hybrid system of Application- Specific Integrated Circuit (ASIC). In general, input always comes from a detector or sensors in more specific word and meanwhile the output goes to the activator which may start or stop the operation of the machine or the operating system.

An embedded system is a combination of both hardware and software. Each embedded system is unique and the hardware is highly specialized in the application domain. Hardware consists of processors, microcontroller, IR sensors etc. On the other hand, Software are just like a brain of the whole embedded system as this consists of the programming languages used which make hardware work. As a result, embedded systems programming can be a widely varying experience. The diagram below shows the general structure of an Embedded System.

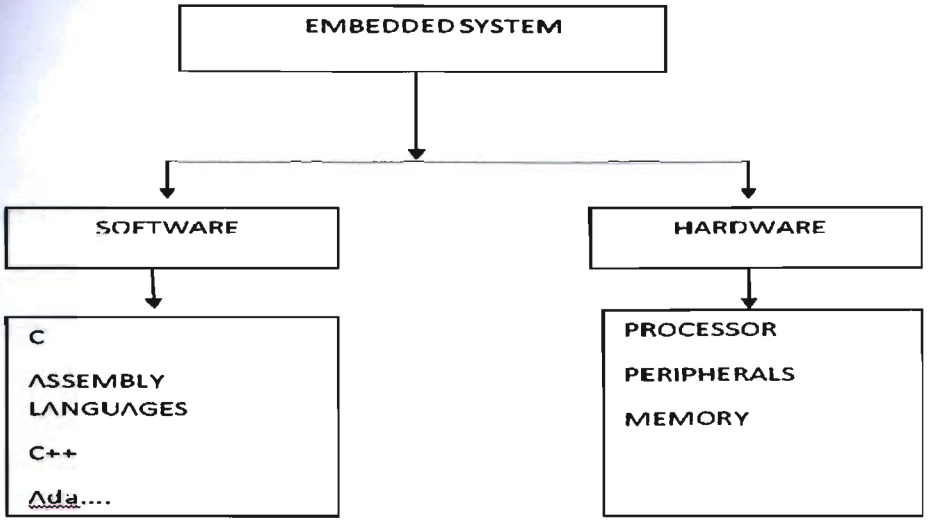


Figure 1.1: Framework Embedded System

1.8 Chapters Layout

The introductory chapter gives a brief explanation of the basic concepts of robotics and a limited sketch of our project. It also discusses the most important part of the device – the microcontroller. There are also some short descriptions on the evolution and history of a very popular microcontroller that we used – Arduino.

Chapter 2: Describes each components we have used and also the theories and analogies behind them.

Chapter 3: Deals with the design and implementation part. This also focuses on the most important part – the circuit design, connection and implementation.

Chapter 4: Here we combine the auto mode and the manual mode and expand the idea of the customizable robot.

Finally **Chapter 5**: Brings to the conclusion and discusses on the final result and the application and future work scope.

Chapter 2

Components Used in the Design

2.1. Introduction

The designed system consists of L293D motor driver, webcam, voltage regulator, RS-232 cable and image processing with microcontrollers and MATLAB which were used as master and slave devices. In this chapter, we are going to discuss briefly about the functions and characteristics of different components used in the system.

2.2. Used Equipment

2.2.1 L293D Motor Driver

L293D is a dual H-bridge motor driver integrated circuit (IC). Motor drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motorsL293D contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions; respectively. Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled. As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state.

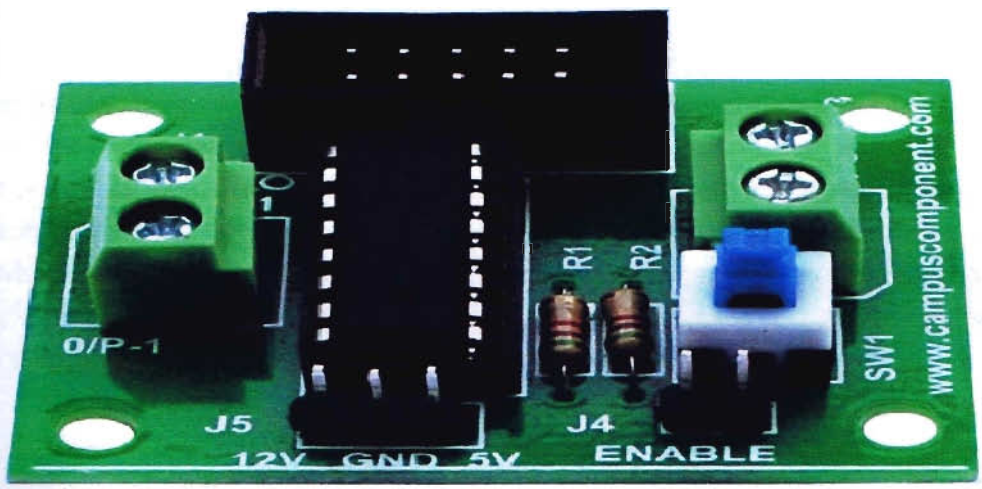


Figure 2.1: L293D Motor Driver

2.2.1.1 Pin Diagram

Figure 2.2: Pin Diagram of Motor Driver

2.2.1.2 Features of Dc Gear Motor

- Easily compatible with any of the system
- Easy interfacing through FRC (Flat Ribbon Cable)
- External Power supply pin for Motors supported
- Onboard PWM (Pulse Width Modulation) selection switch
- 2pin Terminal Block (Phoenix Connectors) for easy Motors Connection
- Onboard H-Bridge base Motor Driver IC (L293D)

2.2.1.3 DC Gear Motor

Geared DC motors can be defined as an extension of DC motor which already had its Insight details demystified here. A geared DC Motor has a gear assembly attached to the motor. The speed of motor is counted in terms of rotations of the shaft per minute and is termed as RPM .The gear assembly helps in increasing the torque and reducing the speed. Using the correct combination of gears in a gear motor, its speed can be reduced to any desirable figure. This concept where gears reduce the speed of the vehicle but increase its torque is known as gear reduction. This Insight will explore all the minor and major details that make the gear head and hence the working of geared DC motor.

2.2.1.4 External Structure of Dc Gear Motor

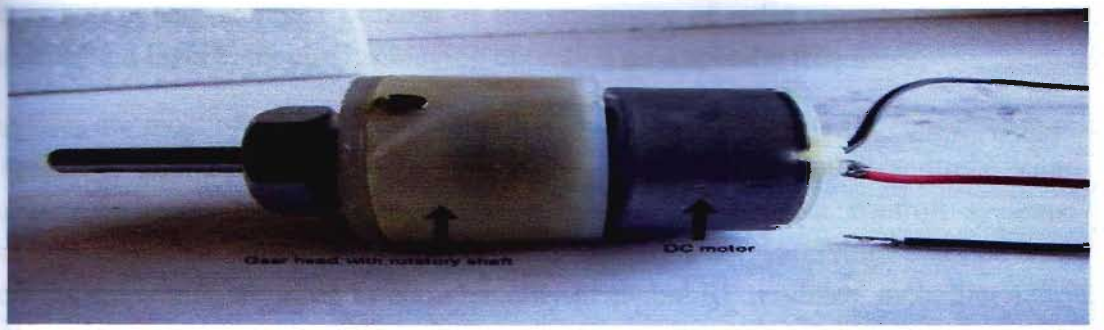


Figure 2.3: DC Gear Motor.

2.2.1.5 Application

Our planetary and spur DC Gear Motors (also known as mini-geared motors and micro-geared motors) are specifically designed for present and future applications. Our Gear Motors are used for various applications in the areas of medicine and healthcare, automobile systems, drive systems, or as a variety of positioning or industrial. These miniature gear motors have also various much-needed characteristics, such as form factors, strength, torque, linear performance and other technical capabilities. All of which are important for such applications. In addition, Precision Micro drives Nano Planetary, Micro Spur, and Miniature meet high quality standards of reliability, ruggedness, and compactness. These are characteristics that we need to use in professional commercial applications.

2.2.2 RS-232 Cable

Information being transferred between data processing equipment and peripherals is in the form of digital data which is transmitted in either a serial or parallel mode. Parallel communications are used mainly for connections between test instruments or computers and printers, while serial is often used between computers and other peripherals. Serial transmission involves the sending of data one bit at a time, over a single communications line. In contrast, parallel communications require at least as many lines as there are bits in a word being transmitted (for an 8-bit word, a minimum of 8 lines are needed). Serial transmission is beneficial for long distance communications, whereas parallel is designed for short distances or when very high transmission rates are required.

RS-232 is defined as the "Interface between data terminal equipment and data communications equipment using serial binary data exchange." This definition defines data terminal equipment (DTE) as the computer, while data communications equipment (DCE) is the modem. A modem cable has pin-to-pin connections, and is designed to connect a DTE device to a DCE device.

2.2.3 RF Module

An RF module (radio frequency module) is a usually small electronic circuit used to transmit or receive radio signals on one of a number of carrier frequencies. RF modules are widely used in electronic design owing to the difficulty of designing radio circuitry. Good electronic radio design is notoriously complex because of the sensitivity of radio circuits and the accuracy of components and layouts required achieving operation on a specific frequency.

RF modules are most often used in medium and low volume products for consumer applications such as garage door openers, wireless alarm systems, industrial remote controls, smart sensor applications, wireless home automation and wireless robotic arm controll systems. In the following figure 4.8 show the RF module's transmitt an receiving site. Several carrier frequencies are commonly used in commercially-available RF modules, including 433.92 MHz, 315 MHz, 868 MHz and 915 MHz These frequencies are used because of national and international regulations governing the use of radio for communication.

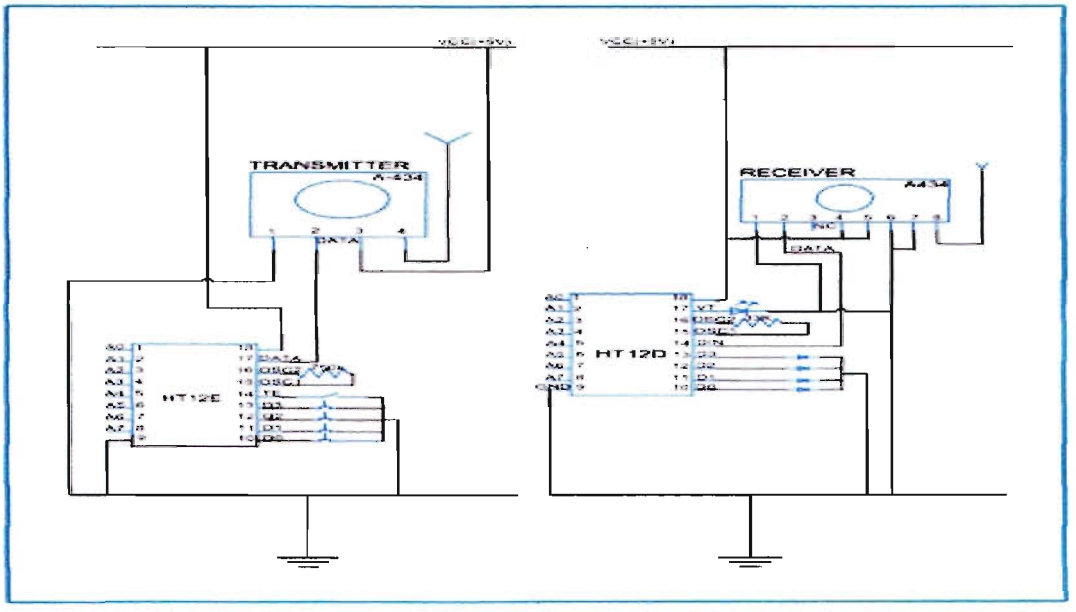


Figure 2.4: Circuit Diagram of RF Module and Receiver.

2.2.3.1 Pin Diagram

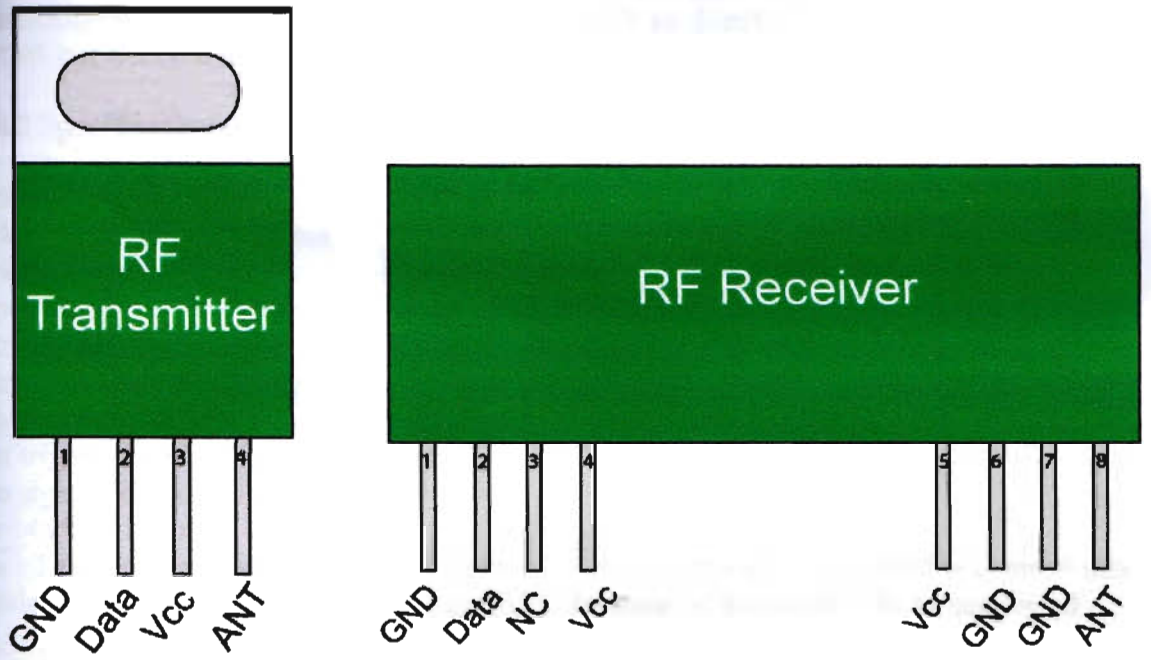


Figure 2.5: Pin Diagram of RF Module.

2.2.4.1 Description

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats or dolphins do. It offers excellent range accuracy and stable readings in an easy-to-use package. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). Similar in performance to the SRF005 but with the low-price of a Sharp

2.2.4.2 Specification

Working Voltage : 5V(DC)
Working Current : max 15 ma
Working frequency : 40HZ
Output Signal : 0-5V (Output high when obstacle in range)
Sentry Angle : max 15 degree
Sentry Distance : 2cm - 500cm
High-accuracy : 0.3cm
Input trigger signal : 10us TTL impulse
Echo signal : output TTL PWL signal
Size : 45*20*15mm
Note : This module is not suitable to connect with electric power, if you need to connect this module with electronic power, then let the GND terminal of this module to be connected first, otherwise, it will affect the normal work of the module

2.2. 4.3 Interface

Pin:
1 : VCC ; 2 : trig (T) ; 3 : echo (R) ; 4 : GND

2.2.4.4 Usage

Supply module with 5V, the output will be 5V while obstacle in range, or 0V if not.
The out pin of this module is used as a switching output when anti-theft module, and without the feet when ranging modules,
Note : the module should be inserted in the circuit before been power, which avoid producing high
level of misoperation ; if not, then power again.
Module Working Principle:
(1) Adopt IO trigger through supplying at least 10us sequence of high level signal,
(2) The module automatically send eight 40khz square wave and automatically detect whether receive the returning pulse signal,
(3) If there is signals returning, through outputting high level
and the time of high level continuing is the time of that from the ultrasonic transmitting to

distance = (high level time * sound velocity (340M/S) / 2

2.2.4.5 Circuit Logic

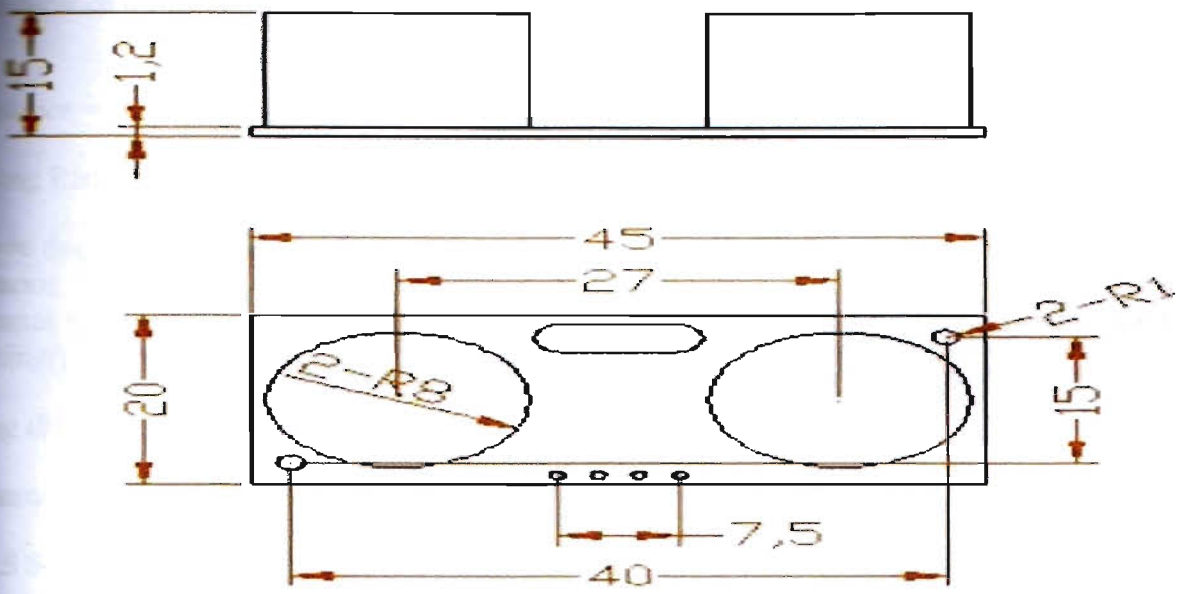


Figure 2.6: Timing Diagram for HC-SR04

2.2.4.6 Using the HC-SR04 with an Arduino

There is an Arduino library for the HC-SR04 that offers two ways to use the sensor. To install, download the "Ultrasonic Library" from this page, unzip the release package into your "arduino-0018/libraries/" folder. Open the Arduino IDE and include the library by Sketch-Import library-Ultrasonic. There is also an example sketch in File-Examples-Ultrasonic-Ultrasonic Demo.

2.2.4.7 Ultrasonic Library

The library includes 3 functions:

1. Ultrasonic(int TP, int EP)

This is a initial function for ultrasonic ranging module, choose the pins for module TRIG and ECHO pin. For example:

Ultrasonic(13,12);

defines the digital pin 13 of Arduino as the TRIG pin of HC-SR04 and pin 12 for the ECHO pin.

2. long Timing()

This function triggers the ultrasonic module and returns the duration that the ECHO pin was held high. For example:

long time; Ultrasonic hcsr; time = hcsr.Timing();

The distance of the object corelates to the time the ECHO pin is held high. The distance formula is:

Distance = ((Duration of high level)(Sonic :340m/s))/2*

3. long Ranging(int sys) — (sys : CM / INC)

If you don't want to change the time into distance yourself, this function will help you get the distance immediately. This function has a parameter (using CM or ICN) that shows the distance in centimeters or inches. This function will call Timing() and you don't need to use the Timing() before it. For example:

long distance; Ultrasonic hcsr; distance = hcsr.Ranging(CM);

Returns the distance in centimeters.

2.2.5 Servomotor

A **servomotor** is a rotary actuator that allows for precise control of angular position, velocity and acceleration.[1] It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.Servomotors are not a different class of motor, on the basis of fundamental operating principle, but use servomechanism to achieve closed loop control with a generic open loop motor. In other words, a servomotor is just a regular motor with a sensor installed, typically to measure angular position during operation.

Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.

2.2.5.1 Mechanism

As the name suggests, a servomotor is a servomechanism. More specifically, it is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is some signal, either analogue or digital, representing the position commanded for the output shaft.The motor is paired with some type of encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signalis generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.The very simplest servomotors use position-only sensing via a potentiometer and bang-bang control of their motor;

the motor always rotates at full speed (or is stopped). This type of servomotor is not widely used in industrial motion control, but it forms the basis of the simple and cheap servos used for radio-controlled models.More sophisticated servomotors measure both the position and also the speed of the output shaft. They may also control the speed of their motor, rather than always running at full speed. Both of these enhancements, usually in combination with a PID control algorithm, allow the servomotor to be brought to its commanded position more quickly and more precisely, with less overshooting.

2.3 Development Boards

2.3.1 Arduino Uno R3 Development Boards

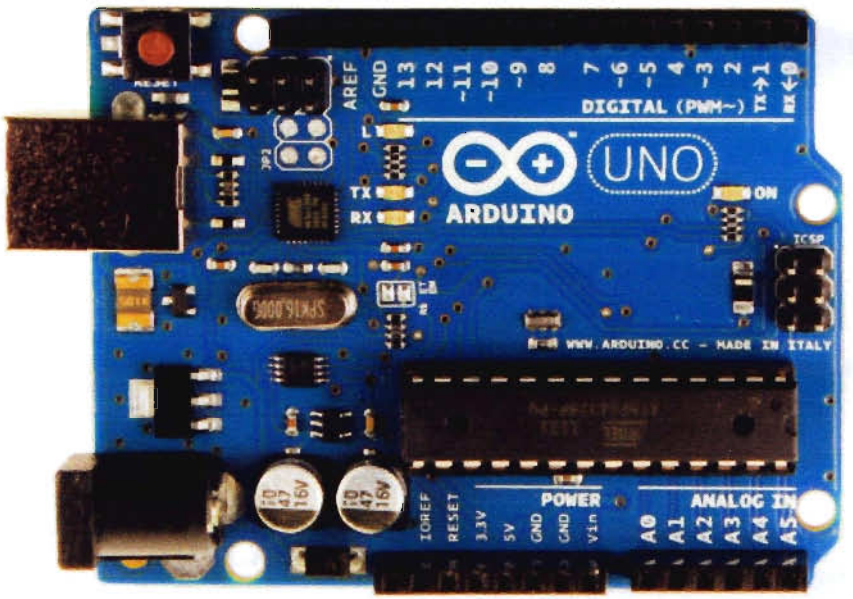


Figure 2.7: The Arduino Uno R3 Board

The Arduino Uno R3 is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

2.3.1.1 Features

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

2.3.1.2 Schematic & Reference Design

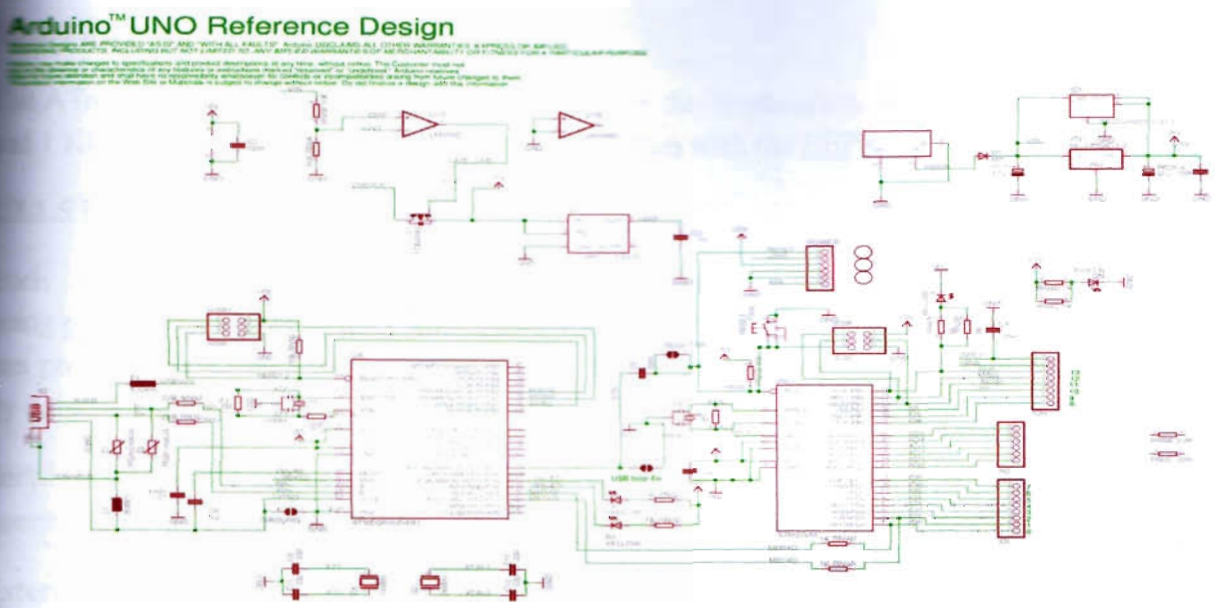


Figure 2.9: Schematic diagram for Arduino Uno R3 Board

2.3. 1 .3 Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

V_{IN} The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it. 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND. Ground pins.

This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

2.3.1.4 Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

2.3.1.5 Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), anddigitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.There are a couple of other pins on the board: Reference voltage for the analog inputs. Used with analogReference(). Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8, 168, and 328 is identical.

2.3.1.6 Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to

software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

2.3.1.7 Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials. The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

we can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.

On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

we can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

2.3.1.8 Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of theATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset

line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

2.3.1.9 USB Over current Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

2.3. 1.10 Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

2.4 Summary

This chapter gave very brief descriptions about each components used for designing purpose. Output devices for displaying purpose like the Computer will be discussed later. The dc gear motor that were used here were for the purpose of verifying the systems working capability and for successful to control the robotic arm using microcontroller. We suggest that it should only be used for learning purpose and should be used for higher study purpose.

Chapter 3

System Design and Development Using Microcontrollers

3.1 Control system with RF module for manual mode

For developing the manual control of our robot we chose cheap RF modules of a transmitter and a receiver pair that called for two different circuitries. One was the transmitter part that we call the remote and the other is the receiver part where the receiver module was connected to the microcontroller. In the remote we used some push buttons each assigned for different tasks. Also, we used two Arduino uno R3 boards separately for the transmitting part and the receiving part.

The connection of the remote goes as below –

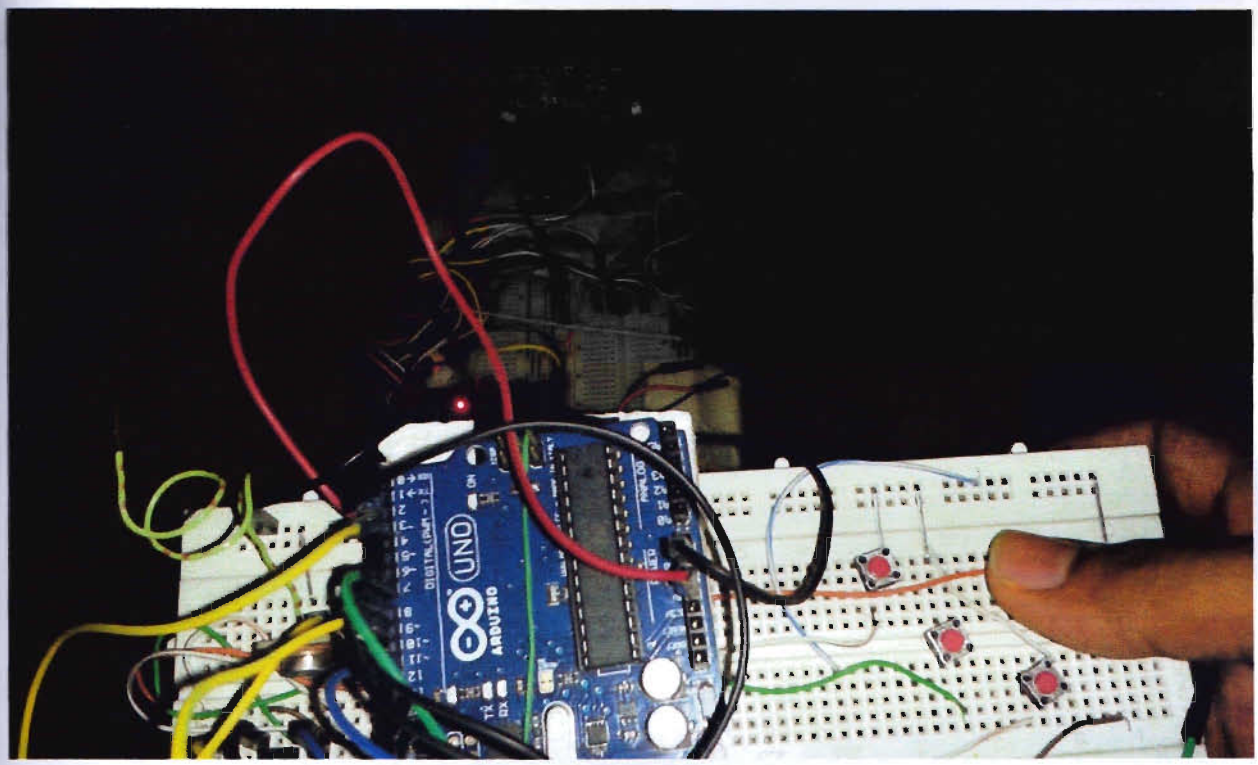


Figure: 3.1 Implementation of manual control

Step 1: We connected 6 push switches in the microcontroller digital pins 7, 8, 9, 2, 11, 12, all set as input.

Step 2: There are 4 pins in the transmitter module. They are V_{cc}, GND, Data and the Antenna pin. The V_{cc} and GND pins were connected respectively to the 5V and the ground pin of the microcontroller. The Data pin was connected to the microcontroller pin 3. And the antenna pin of the transmitter module was just connected to a wire used as the antenna.

The connection part of the receiver end goes as below –

Step 1: We placed the receiver module beside the Arduino board.

Step 2: There are 8 pins in the receiver module. They are two V_{cc}, three GND pins, a Data pin, a DER pin and an Antenna pin. After connecting the V_{cc} and the GND pins as usual we connected the Data pin to microcontroller pin 8, DER pin to 5v power supply and a simple wire to the Antenna pin working as an antenna.

Step 3: The Arduino was powered with 12V rechargeable dc batteries.

3.2 Summary

The chapter discussed about the robot controlled by the microcontroller where the functionality of the system was verified. The GUI couldn't be created due to lack of time as the project submission deadline had to be met as part of the "Project and Thesis" course. Besides this, all the objectives of the robot control system have been accomplished and verified from the snapshots provided here. This was done to verify all the functionalities of the system and that it was working properly.

Chapter 4

Combination of Auto and Manual Mode

4.1Introduction

Until now we just designed the Auto part of the robot. It was time to merge the Auto control mode with the manual control mode. It was a big challenge because it started to turn out a lot more complicated when we started to combine connections, programs and logics together. To make things worse with our utter unfortunate we discovered that the VirtualWire library that was needed for the RF control programming wouldn't work together with the Servo library because they both shared the same Timer1 clock source. Now, that was a problem. We had to find a solution by editing one library function and divert its time source to Timer2. We picked up the servo library and changed it to the way that it would use the Timer2 instead. Well, that option left us with a limitation that we could only use 8 servos instead of 16, but that was not our problem and most importantly after the library editing both started to work well together. However, let us learn how we managed to combine the whole into one. Below there are the logic diagrams for both the transmitter part and the receiver part.

4.2 Logic Diagram

4.2.1 Flow chart for the transmitter part

Here is the flow chart of the logic we used for the transmitter part-

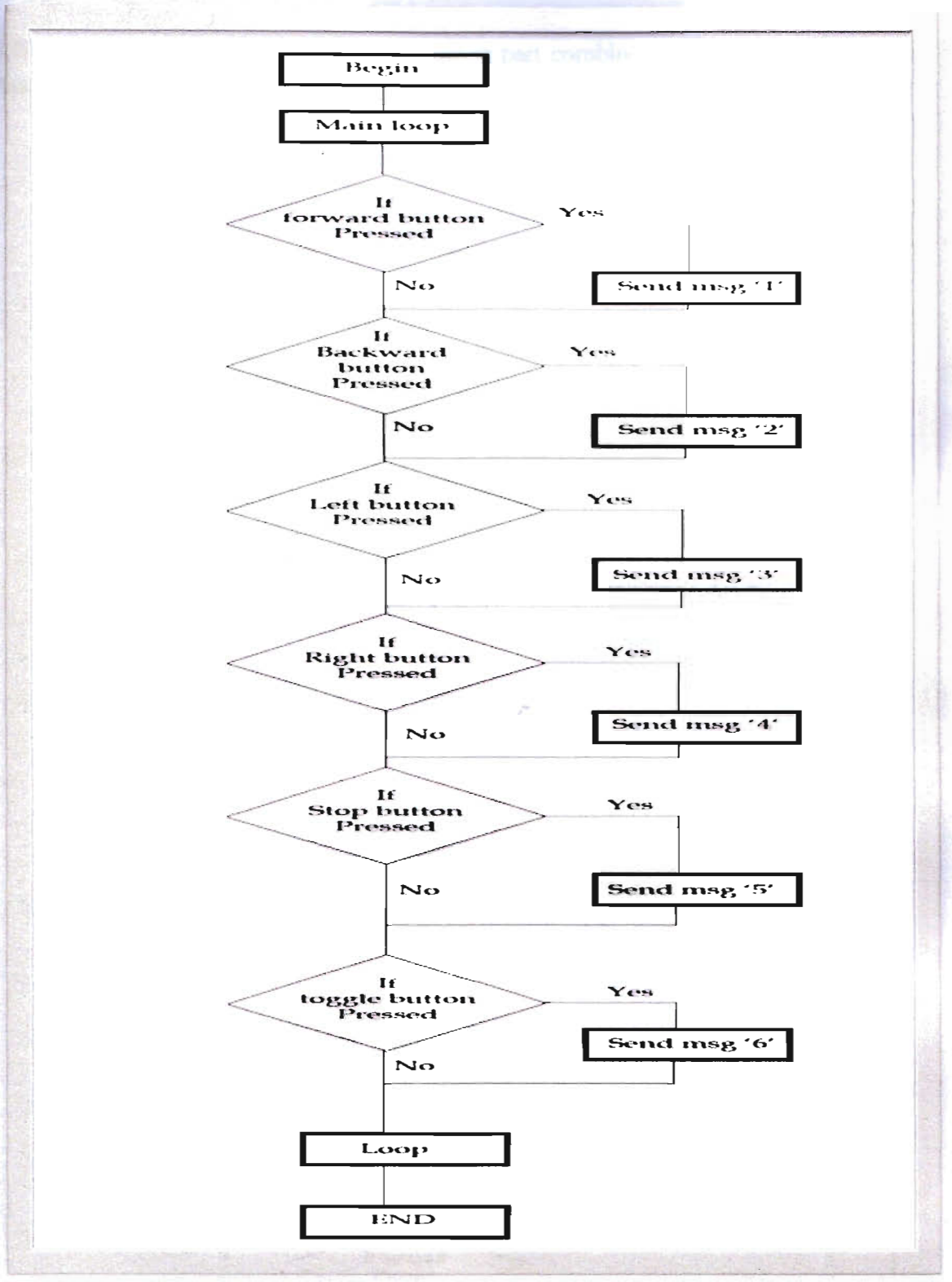
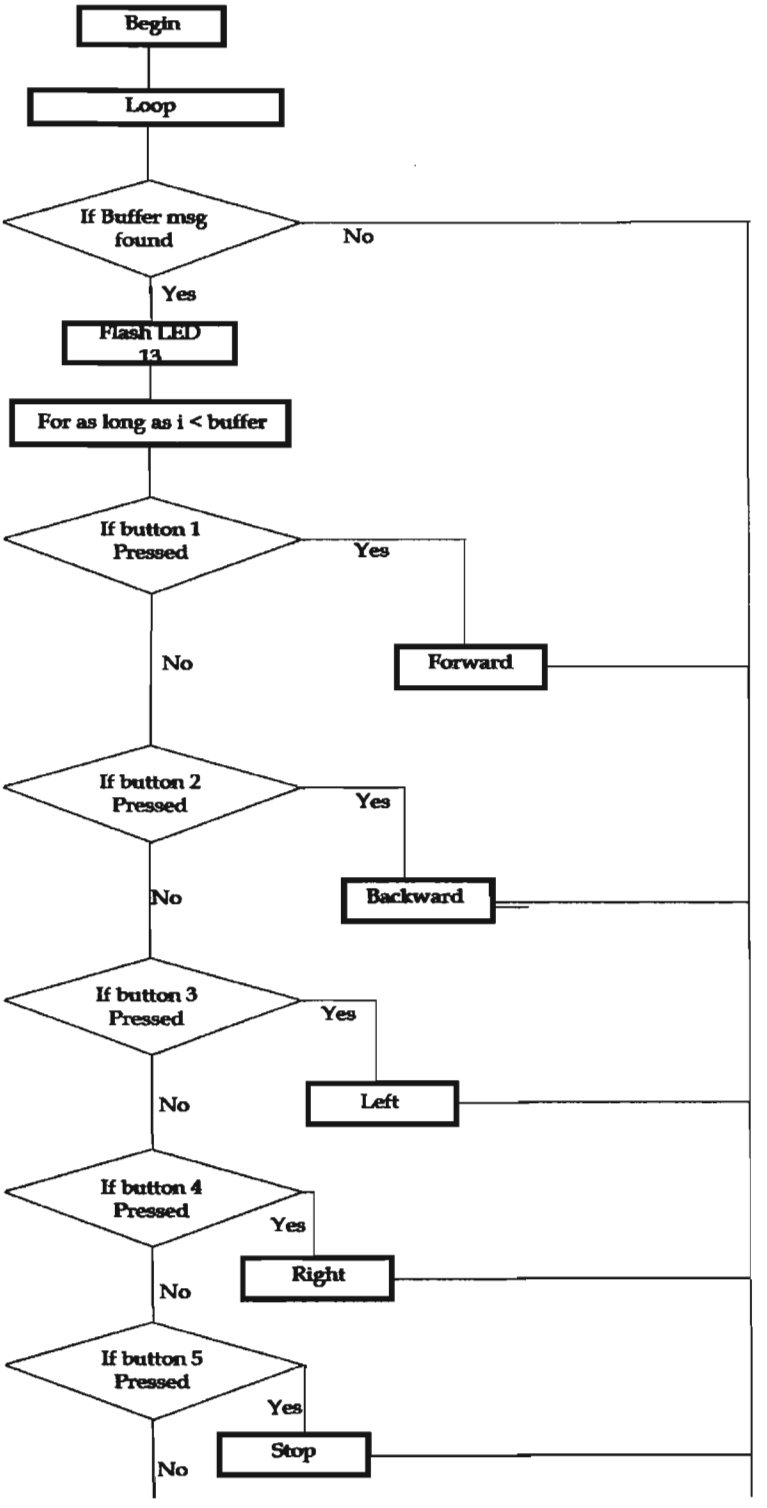
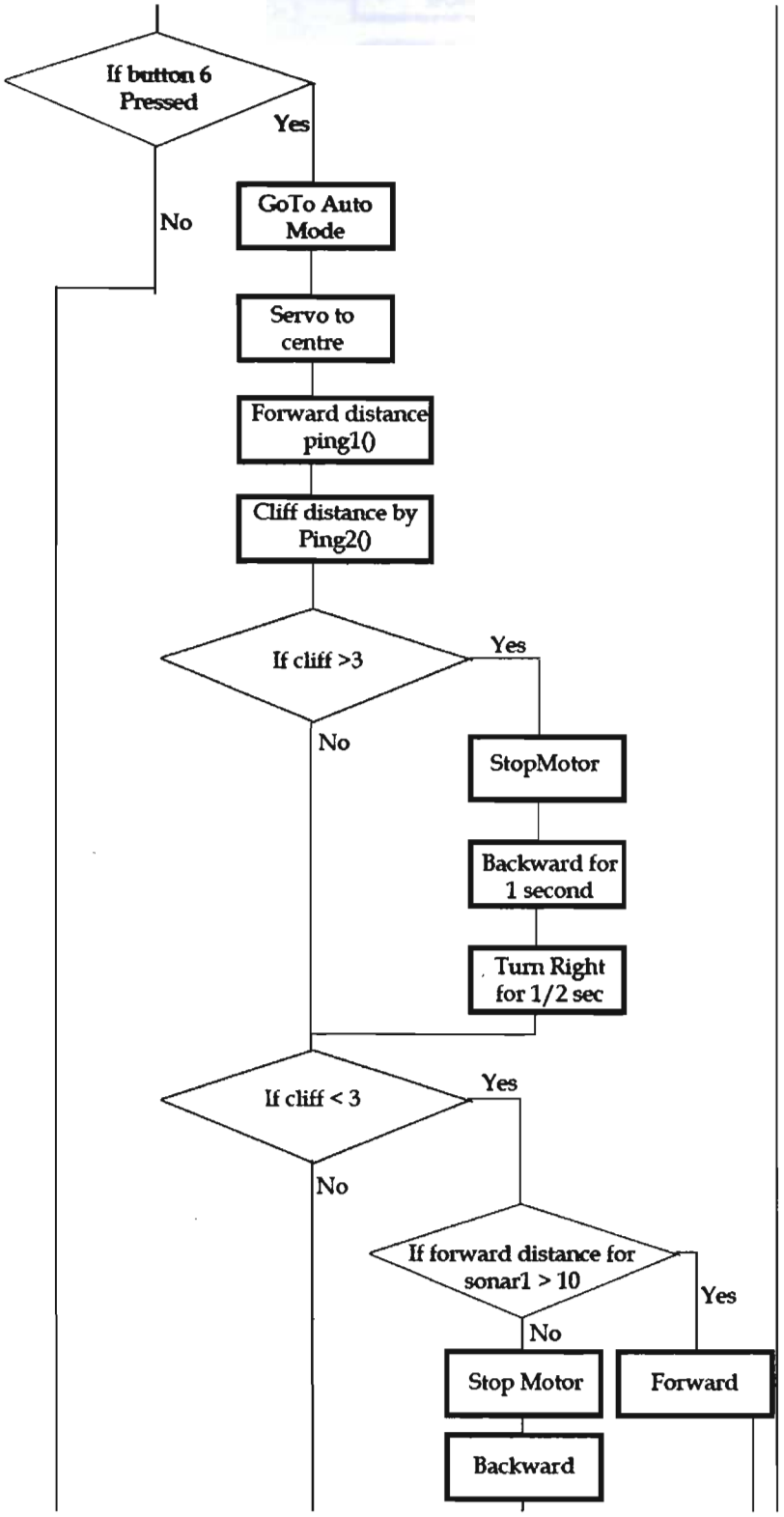


Figure: 4.1 Logic diagram for the transmitter part

4.2.2 Logic diagram for the receiver part

The flow chart for the logic used for the receiver part combined with both the Auto mode and Manual mode goes as follow




If button 6 Pressed
Yes
No
GoTo Auto Mode
Servo to centre
Forward distance ping1()
Cliff distance by Ping2()
If cliff >3
Yes
No
StopMotor
Backward for 1 second
Turn Right for 1/2 sec
If cliff < 3
Yes
No
If forward distance for sonar1 > 10
Yes
No
Stop Motor
Forward
Backward

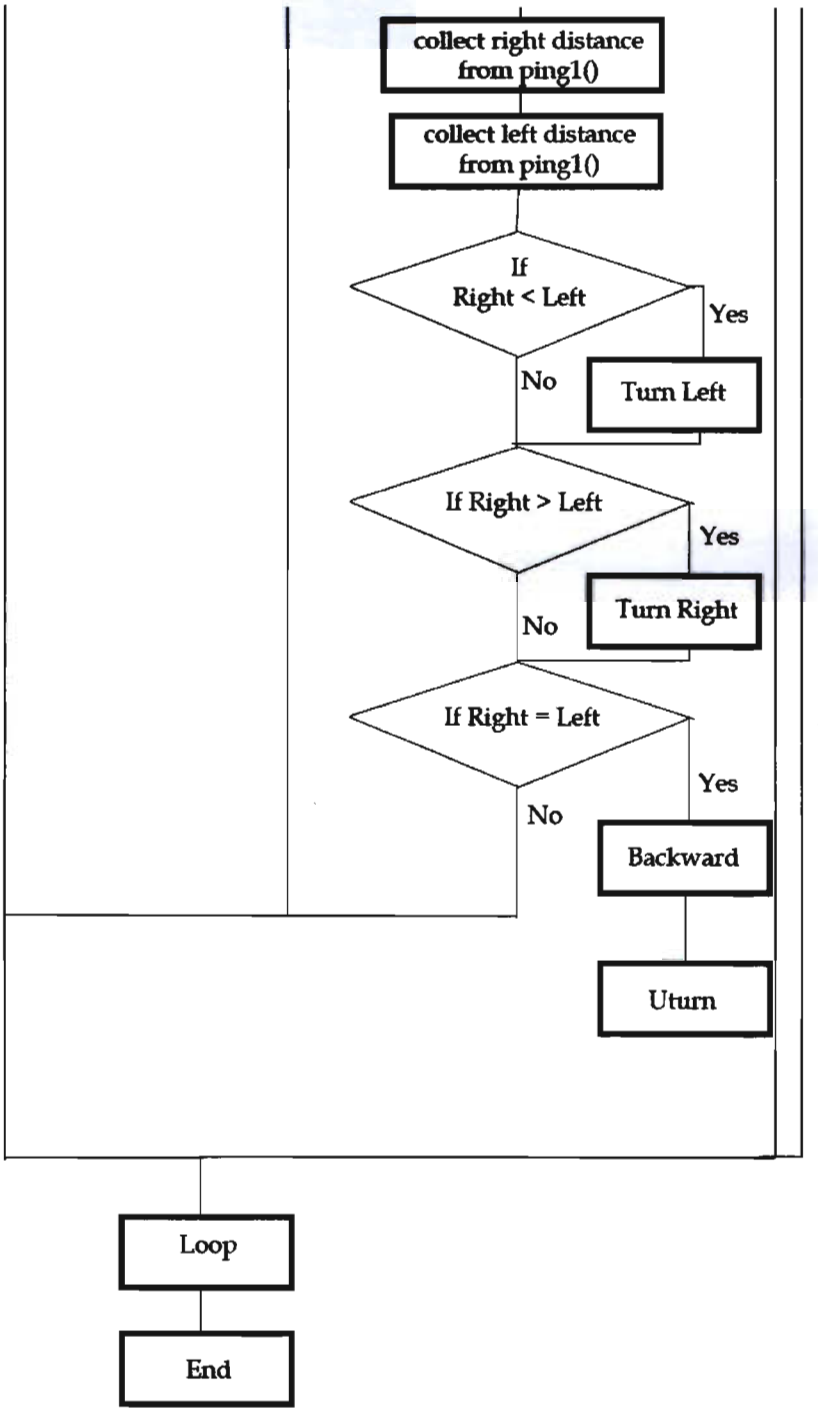


Figure: 4.2 Flow chart of the logic diagram of the receiver part

4.3 Conclusion

This works in the way that, the user has a toggle button in the remote, he can choose between the Auto mode and the manual mode by turning it 'on' or 'off'. When it is triggered in the manual mode then the user can control it with the 'forward', 'backward', 'left', 'right' and 'stop' switches. And when he switches to the auto mode the robot doesn't need to be controlled by others any longer, it can roam freely by itself avoiding all the collisions and cliffs. Although it is in its free move but user can still control it with other switches. This whole combination gives the user the super control and the flexibility to use it.

Chapter 5

Discussions

5.1 Discussions

In this project, a robust system is developed that can be controlled wirelessly from distance and when necessary it can be left alone in the Auto mode and it can roam freely with no risk from falling off from cliffs or getting stuck with any static or moving obstacles. Such a device is very handy for many tasks. Initially in this stage it can be used in any scientific expedition where there is no or very little access for human beings. Or even in the places where human entry is so dangerous that can cost life such as the Coal mines, inside the volcano or in the radiated areas. It can carry things in distances by only adding live cam transmission for proper controlling in greater distances. These are just the possible applications for its initial stage. But our main concept is to use it as an ultra customizable robot. It can be expanded and loaded with a lot of features that can be customized on user demand. So, let us demonstrate the concept for you. Below we see the sample features that we can add or remove for users of different needs.

5.2 For research purpose

We can add GSM control, wireless video transfer through camera on top of our already developed control systems for users who may need it for research. We can also add sensors like weather, humidity sensor, heat sensor, proximity sensor, accelerometer etc. User may choose which ones they need, there may be a limitation on number of features depending on the hardware limitation but we hope as the project develops, hardware will get updated too with better capacity and more expanded ability. But anyway, this will make up for a decent robot for research purpose.

5.3 For home security

For anyone who might be interested to use it as a home security system, we may provide him auto cleaner + SMS update of home status based on different sensors etc.

5.4 For Industrial purpose

For industries we are hopeful to add robotic arms to our system which will also include a wireless video transmission feature and line following ability. So the thing is that it will be ready for industrial usage with features like, it could be an autonomous robot that could repeat predefined tasks over and over again and also be a manually controllable robot. With robotic arms it will definitely be a highly useable device for industries.

5.5 For defense and warfare

It is also possible to use it for defense and warfare scenarios. If we can add a weapon control system which is fortunately already available to add, it will be a great addition for the case of defense and warfare. With an extra feature as auto status update on radar and base system it will be a deadly land drone for defense and Military department.

Chapter 6

Conclusions and future work scope

6.1 Suggestion for future work

The applications stated above are some demo applications that are absolutely possible with its future development. Initially for the limitation of time and required fund we were able to develop just the basic control system for the robot we imagined. These features yell the strong future work scope and possible development of this project. We have named our robot as the 'PRO-TRON1', so we are hopeful to come up with Pro-Tron2, 3, 4... in the future. Initially we would like to make all the features available first. Then we will be able to present our 'PRO-TRON' as a very powerful industrially customizable robot.

6.2 Difficulties that we faced

First of all, we all started from the ground zero level with no knowledge on anything. So, literally, it was very hard for us to advance through different stages. There were many limitations we faced, ranging from equipment selection to the implementation process. To name a few, the first problem we encountered was the insufficient resource and inefficient equipments in the local market in Bangladesh. We couldn't find a useable chassis in the market, so we had to design one on our own. Another notable problem occurred when we tried to combine our multiple control system. As stated earlier, we had to find the root of the problem when servo library refused to work with virtualwire library. Fortunately we solved it by exchanging the default servo library with ServoTimer2 in Arduino programmer which uses Timer2 instead of Timer1. The next problem was that, the ServoTimer2 would accept servo angle in milliseconds instead of degrees, so we had to go through the whole program changed accordingly.

6.3 Conclusion

However, we are happy to complete the project successfully against all odds. We sorted out all the problems and were fortunately able to fix them all. As we were successful to achieve our primary goals, we are opting for the future advancement of the robot. We hope that, we will be able to complete all the features needed for its extreme applications.

REFERENCES

[1] 2R Hardware & Electronics: Arduino Uno. 2013. 2R Hardware & Electronics: Arduino Uno. [ONLINE] Available at: http://2r-he.blogspot.fi/2011/04/arduino-uno.html. [Accessed 16 May 2013].

[2] Application-Specific Integrated Circuits (ASIC's). 2013. Application-Specific Integrated Circuits (ASIC's). [ONLINE] Available at: http://www.siliconfareast.com/asic.htm. [Accessed 19 July 2013].

[3] Arduino - Software. 2013. Arduino - Software. [ONLINE] Available at: http://arduino.cc/en/Main/Software. [Accessed 16 May 2013].

[4] Arduino - HomePage . 2013. Arduino - HomePage . [ONLINE] Available at: http://arduino.cc/. [Accessed 16 May 2013].

[5] Arrick Robotics - Comparing A Robot and BOE-Bot . 2013. Arrick Robotics - Comparing ARobot and BOE-Bot . [ONLINE] Available at: http://www.arrickrobotics.com/arobot/arobotboebot.html. [Accessed 15 October 2013].

[6] Banzi, Massimo, 2009. Getting Started with Arduino. 1st ed. Farnham: O'Reilly Media.

[7] BS2 programming board (board of education). 2014. BS2 programming board (board of education). [ONLINE] Available at: http://users.ntua.gr/dpiperid/MyWebPage/Contructions/Bs2BoardEN.htm. [Accessed 20 January 2014].

[8] Embedded Systems Design – Heath Steve - Google-kirjat. 2013. Embedded Systems Design - Steve Heath - Google-kirjat. [ONLINE] Available at:=true. [Accessed 18 July 2013].

[9] Free Merriam-Webster Dictionary. 2013. y. [ONLINE] Available at: http://www.merriam-webster.com/dictionary/robot. [Accessed 18 August 2013].

[10] History of Computers and Computing, Birth of the modern computer, The bases of digital computers, Integrated Circuit. 2014. History of Computers and Computing, Birth of the modern computer, The bases of digital computers, Integrated Circuit. [ONLINE] Available at: http://history-computer.com/ModernComputer/Basis/IC.html. [Accessed 31 January 2014].

[11] http://arduinobasics.blogspot.com/2012/11/arduinobasics-hc-sr04-ultrasonic-sensor.html

[12] Infrared-based obstacle avoider- AIEPIC Project 2009-2010. 2013. Infrared based obstacle avoider- AIEPIC Project 2009-2010. [ONLINE] Available at: http://www.siliconindia.com/aiepic/project/infrared_based_obstacle_avoider-pid=8419.html. [Accessed 10 November 2013].

[13] Introduction To Embedded Systems. 2013. Introduction To Embedded Syestes. [online] Available at: http://www.slideshare.net/yayavaram/introduction-to-embedded-systems-2614825. [Accessed 21 July 2013].

[14] Obstacle Avoiding Robot Report Robot23. 2013. Obstacle Avoiding Robot Report Robot23. [ONLINE] Available at: http://www.slideshare.net/abhi230789/obstacle-avoiding-robot-report-robot23. [Accessed 20 November 2013].

[15] Obstacle Detctor Robot. 2013. Obstacle Detctor Robot. [ONLINE] Available at: http://www.slideshare.net/NikitaKaushal/obstacle-detctor-robot. [Accessed 21 July 2013].

[16] Obstacle Detection and avoidence Robot (1) Computer Science Project Ideas. 2013. Obstacle Detection and avoidence Robot (1) Computer Science Project Ideas. [ONLINE] Available at: http://seminarprojects.com/Thread-obstacle-detection-and-avoidance-robot#ixzz2lvDgFg7F. [Accessed 10 November 2013].

[17] Oxford dictionary (British & World English). 2013. [ONLINE] Available at: http://www.oxforddictionaries.com/definition/english/robot. [Accessed 18 August 2013].

[18] The Making of Arduino - IEEE Spectrum. 2013. The Making of Arduino - IEEE Spectrum. [ONLINE] Available at: http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino. [Accessed 16 May 2013].

[19] Webopedia Computer Dictionary. 2013.. [ONLINE] Available at: http://www.webopedia.com/TERM/I/integrated_circuit_IC.html. [Accessed 3 August 2013].

[20] Webopedia Computer Dictionary. 2014.. [ONLINE] Available at: http://www.webopedia.com/TERM/M/microcontroller.html. [Accessed 20 January 2014].

Wheat, Dale, 2011. Arduino Internals. 1st ed. New York: Apress.

Appendix A

Programming code for transmitter (Tx)

```
#include <VirtualWire.h>
//Assigning controller buttons to Digital Pins
int toggle = 7;
int forward = 8;
int backward = 9;
int rightTurn = 2;
int leftTurn = 11;
int stopMotor = 12;
int remotePins[]= {7,8,9,2,11,12};//array to store pin no
void setup()
{
  Serial.begin(9600);          // Debugging only
  Serial.println("setup");
  // Initialise the IO and ISR
  vw_setup(2000);    // Bits per sec
  vw_set_tx_pin(3); //Transmitter Data Pin to Digital Pin 3

  for (int i = 0; i<6 ; i++)
  {
   pinMode(remotePins[i], INPUT);
   digitalWrite(remotePins[i],HIGH);
  }
```

```
}
void loop()
{
 char *msg;
 if(digitalRead(forward) == LOW)//if the forward button is pressed
 {
  char *msg = "1";//send 1 to the receiver
  digitalWrite(13, true); // Flash a light to show transmitting
  vw_send((uint8_t *)msg, strlen(msg));//send the byte to the receiver
  vw_wait_tx(); // Wait until the whole message is gone
  digitalWrite(13, false);
 }
 else if(digitalRead(backward) == LOW)//if the back button is pressed
 {
  char *msg = "2";///send 2 to the receiver
  digitalWrite(13, true); // Flash a light to show transmitting
  vw_send((uint8_t *)msg, strlen(msg));//send the byte to the receiver
  vw_wait_tx(); // Wait until the whole message is gone
  digitalWrite(13, false);
 }
 else if(digitalRead(leftTurn) == LOW)//if the left button is pressed
 {
  char *msg = "3";//send 3 to the receiver
  digitalWrite(13, true); // Flash a light to show transmitting
```

```
 vw_send((uint8_t *)msg, strlen(msg));//send the byte to the receiver
 vw_wait_tx(); // Wait until the whole message is gone
 digitalWrite(13, false);
}
else if(digitalRead(rightTurn) == LOW)//if the right button is pressed
{
 char *msg = "4";//send 4 to the receiver
 digitalWrite(13, true); // Flash a light to show transmitting
 vw_send((uint8_t *)msg, strlen(msg));//send the message to the receiver
 vw_wait_tx(); // Wait until the whole message is gone
 digitalWrite(13, false);
}

else if(digitalRead(stopMotor)==LOW)//if the stop button is pressed
{
 char *msg = "5";//send 5 to the receiver
 digitalWrite(13, true); // Flash a light to show transmitting
 vw_send((uint8_t *)msg, strlen(msg));//send the message to the receiver
 vw_wait_tx(); // Wait until the whole message is gone
 digitalWrite(13, false);
}
else if(digitalRead(toggle) == LOW)
 {char *msg = "6";
 digitalWrite(13, true); // Flash a light to show transmitting
  vw_send((uint8_t *)msg, strlen(msg));//send the byte to the receiver
```

```
  vw_wait_tx(); // Wait until the whole message is gone
  digitalWrite(13, false);
 }

}
```

Appendix B

Programming code for receiver (Rx)

```
#include <ServoTimer2.h>
#include <VirtualWire.h>
#define trigPin1 13            // obstacle detector sonar *sonar 1*
#define echoPin1 12
#define trigPin2 10            // cliff detector *sonar2*
#define echoPin2 11
ServoTimer2 myservo;
int en1 = 2;
int en2 = 3;
int in1 = 4;
int in2 = 5;
int in3 = 6;
int in4 =7;
int motorPin[] = {2,3,4,5,6,7};
float leftDistance1;
float rightDistance1;
float Duration1;
float Distance1;
float Duration2;
float Distance2;
void setup()
```

```
{
  Serial.begin(9600);          // starts the serial monitor
  Serial.println("setup");
  vw_setup(2000);     // Bits per sec
  vw_set_rx_pin(8);    //Rx Data pin to Digital  Pin 8
  vw_rx_start();       // Start the receiver
  myservo.attach(9);
pinMode(trigPin1, OUTPUT);
pinMode(echoPin1, INPUT);
pinMode(trigPin2, OUTPUT);
pinMode(echoPin2, INPUT);
   for (int j = 0; j < 6; j++)
  {
      pinMode(motorPin[j], OUTPUT);
  }
}

void loop()
{
    uint8_t buf[VW_MAX_MESSAGE_LEN];
    uint8_t buflen = VW_MAX_MESSAGE_LEN;
    if (vw_get_message(buf, &buflen)) // Non-blocking
      {
       digitalWrite(13, true); // Flash a light to show a good message is received
          for (int i = 0; i < buflen; i++)
```

```
{
 Serial.print(buf[i]);//print received command
switch (buf[i])
{

 case '1':
 forward();//go forward
 Serial.println(" = forward");
 break;

 case '2':
 backward();//go backward
 Serial.println(" = backward");
 break;

 case '3':
 left();//go left
 Serial.println(" = left");
 break;

 case '4':
 right();//go right
 Serial.println(" = right");
 break;
```

```
case '5':
stopMotor();  //  stop/brake
Serial.println(" = stopped");
break;
case '6':
if (buf[i] == '6')
 {
   myservo.attach(9);
   myservo.write(1500);   //initialise at centre
   delay(15);
   float Distance1Fwd = ping1();
   float cliff = ping2();

    if (cliff >3)
    {
     stopMotor();
     backward();
     delay(1000);
     right();
     delay(500);
    }

    else if (cliff <3)
    {if (Distance1Fwd > 10)  // no obstacle within 10 cm
```

```
{
  myservo.attach(9);
  forward();
  myservo.detach();
}

else
{
 stopMotor();
 backward();
 delay(100);
 myservo.attach(9);
 myservo.write(910); // turn right
 delay(15);
 rightDistance1 = ping1(); // check Distance1
 delay(500);
 myservo.detach();
 myservo.attach(9);
 myservo.write(2300); //turn left
 delay(200);
 leftDistance1 = ping1(); // check Distance1
 delay(500);
 myservo.detach();
 myservo.attach(9);
 myservo.write(1500);
```

```
        delay(150);
        myservo.detach(); // for avoiding servo loop
        compareDistance1();

      }
     }
              } // if buf[i] ends
   break;

     } //switch ends

         Serial.print(" ");
                  } //for ends
  Serial.println("");
digitalWrite(13, false);
                  } //if ends

} // main loop closes

// All the functions used -
```

```
float ping1()
{digitalWrite(trigPin1, LOW);
delayMicroseconds(2);
digitalWrite(trigPin1, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin1, LOW);  // sonar1 sonic burst
Duration1 = pulseIn(echoPin1, HIGH);
Distance1 = (Duration1/2)/29.1;
return Distance1;}

float ping2()
{digitalWrite(trigPin2, LOW);
delayMicroseconds(2);
digitalWrite(trigPin2, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin2, LOW);  // sonar2 sonic burst
Duration2 = pulseIn(echoPin2, HIGH);
Distance2 = (Duration2 /2)/29.1;
return Distance2;}

void forward()
{digitalWrite(en1, HIGH);
digitalWrite(en2, HIGH);
```

```
digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);
digitalWrite(in3, HIGH);
digitalWrite(in4, LOW);
}

void backward()
{digitalWrite(en1, HIGH);
digitalWrite(en2, HIGH);
digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);
digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);
}

void left()
{digitalWrite(en1, HIGH);
digitalWrite(en2, HIGH);
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
digitalWrite(in3, HIGH);
digitalWrite(in4, LOW);
}

void right()
```

```
{digitalWrite(en1, HIGH);
digitalWrite(en2, HIGH);
digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);
digitalWrite(in3, LOW);
digitalWrite(in4, LOW);
}

void stopMotor()
{digitalWrite(en1, HIGH);
digitalWrite(en2, HIGH);
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
digitalWrite(in3, LOW);
digitalWrite(in4, LOW);
}

void uturn()
{digitalWrite(en1, HIGH);
digitalWrite(en2, HIGH);
digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);
digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);
}
```

```
void compareDistance1()
{ if(rightDistance1 < leftDistance1) //if left is free
{left();                    // turn left
delay(500);}
else if(rightDistance1 > leftDistance1) //if right is free
{right();
delay(500);
}
else      //if both are crowded
{backward();
  delay(1000);
  uturn();
  delay(3300);
}
}
```