

GestureIt

A PROJECT REPORT

Submitted by

Rishabh Verma (120101195)

Rishi Chandak (120101197)

Saurabh Saluja (120101218)

in partial fulfillment for the award of the degree

of

Bachelor of Technology

IN

Computer Science Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING AND TECHNOLOGY

SHARDA UNIVERSITY, GREATER NOIDA-201306

APRIL 2016

SHARDA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING AND TECHNOLOGY

GREATER NOIDA



Project Report

On

“GestureIt”

Submitted in partial fulfillment for the

Award of degree of

Bachelor of Technology

Batch 2012-2016

In

Computer science & Engineering

Submitted to: -

Dr. Ishan Ranjan

HOD, CSE

Project Guide: -

Ms. Jyotsna Seth

(Asst. Professor CSE)

Submitted by:-

Rishabh Verma

Rishi Chandak

Saurabh Saluja

Group No. G-63

DECLARATION

I hereby declare that this project work submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place:

Signature of the Student

Date

Name:

BONAFIDE CERTIFICATE

Certified that this project report titled “**GestureIt**” is the bonafide work of Mr. Rishabh Verma, Mr. Rishi Chandak and Mr. Saurabh Saluja who carried out project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Signature of Head of Department

Name: Dr. Ishan Ranjan

Signature of Supervisor

Name:
Designation:

Place: Greater Noida

Date:

Signature of External Examiner

Date:

ABSTRACT

Purpose of GestureIt is to bridge the gap between humans and the systems they interact with. To achieve this goal our group has designed and build a Gesture Based Hardware Control System which will be controlled through gestures of our hands. Many Input methods have been devised: keyboards, mice, joysticks, game controllers, and touch screens. But unfortunately, none of these devices remove the barrier between man and machine. With the Gesture Based Hardware Control system, we aim to remove this obstruction by allowing the user to control a hardware device using natural gestures. The Gesture Based Hardware Control takes advantage of a multitude of sensors to capture hand movements and uses this information to control a device – in this case, a modified Remote Controlled car. First we had to learn the interface and architecture of an Arduino board. Second we need to learn how to code on the embedded system and flash the program onto the main Arduino Due and Arduino Pro Mini microcontroller. Following success in stabilizing and working on each and every module, the final stage was to assemble the parts. Once fully built, we need to establish wireless communication between both the microcontrollers using Xbee wireless module.

ACKNOWLEDGEMENT

A major project is a golden opportunity for learning and self-development. We consider our self very lucky and honored to have so many wonderful people lead us through in completion of this project.

First and foremost we would like to thank Dr. Ishan Ranjan HOD CSE who gave us an opportunity to undertake this project.

My grateful thanks to Prof. Jyotsna Seth and Prof. Sachin Sharma for his guidance in my project work “**GestureIt**” who in spite of being extraordinarily busy with academics, took time out to hear, guide and keep us on the correct path. We do not know where we would have been without his help.

CSE department monitored our progress and arranged all facilities to make life easier. We choose this moment to acknowledge their contribution gratefully.

Name and signature of Students

Rishabh Verma (120101195)

Rishi Chandak (120101197)

Saurabh Saluja (120101218)

Table of Contents

CHAPTER-1: PROJECT INTRODUCTION.....	1-4
1.1: Motivation.....	1
1.2: Overview.....	1
1.3: Expected outcome.....	1
1.4: Existing System/Application.....	1
1.5: SRS.....	2
CHAPTER-2: METHODOLOGY.....	5-14
2.1: Overall Descriptions.....	5
2.2: System components & Functionalities.....	5
CHAPTER-3: DESIGN CRITERIA.....	15-17
3.1: System Design	15
3.2: Design Diagrams	16
3.3: System Architecture.....	17
CHAPTER-4: DEVELOPMENT & IMPLEMENTATION.....	18-24
4.1: System modules and flow of implementations.....	18
4.2: Critical Module.....	24
CHAPTER-5: RESULTS & TESTING.....	25-26
5.1: Result.....	25
5.1.1: Success cases.....	25
5.1.2: Failure cases.....	25
5.2: Testing.....	25
5.2.1: Type of testing adapted.....	25
5.2.2: Conclusion of Testing.....	26
CHAPTER-6: CONCLUSION & FUTURE IMPROVEMENTS.....	27-28
6.1: Performance Estimation.....	27
6.2: Usability of Product / system.....	27
6.3: Limitations.....	28
6.4: Scope of Improvement.....	28
REFERENCES.....	29

List of Figures

Fig 2.1: Arduino DUE Microcontroller.....	6
Fig 2.2: Arduino pro mini Microcontroller.....	8
Fig 2.3: Flex Sensor.....	9
Fig 2.4: Force Sensor.....	9
Fig 2.5: Accelerometer.....	10
Fig 2.6: L293D IC.....	11
Fig 2.7: L293D IC Pin Diagram.....	11
Fig 2.8: Xbee Module.....	12
Fig 2.9: DC Motor.....	12
Fig 2.10: Xbee Shield.....	13
Fig 2.11: Xbee USB Adapter.....	14
Fig 3.1: Schematic diagram of the Main Circuit.....	15
Fig 3.2: Flowchart of GestureIt.....	16
Fig 3.3: System Architecture for GestureIt	17
Fig 4.1: Circuit Diagram for GestureIt.....	24
Fig 5.1: Screen Monitor Output 1.....	25
Fig 5.2: Screen Monitor Output 2.....	26
Fig 6.1: Smart glove (Back)	27
Fig 6.2: Smart glove (Front)	27
Fig 6.3: Smart car.....	28

CHAPTER 1

1. PROJECT INTRODUCTION

1.1 Motivation

The old way of thinking: machines are there to do the sorts of work that users didn't want to do, like a dishwasher or a Roomba. The new way of thinking: machines are here to help us do the things that we want to do, but are not able to, like a construction yard crane. If users could reach a level of comfort with machines, as though using them was no more than using an extension of their own bodies, perhaps they could get to a point where they thought of themselves as the ones performing the action. That level of comfort and familiarity is not something that can be accomplished overnight, it would take years, perhaps even generations to reach. Though the change may seem rather daunting, it has to start somewhere, this is how we got motivated to make a glove to control a RC Car.

1.2 Overview

The most common thing that people do when frustrated with devices is perform hand gestures and try to show the machine what they wanted it to do. A controller is unnatural and requires getting used to it on the other hand body language and hand gestures however are instinctual. The GestureIt system utilizes various sensors to capture those hand gestures and interpret them as inputs. A touch of the thumb, a waggle of the finger, or a tilting of the hand all act as control inputs. By utilizing the gestures that most make out of simple habit, the hope is that users will forget the control glove is there at all. The device seems to do what the user wants, as though a direct line of communication exists between the user and the hardware to be controlled. GestureIt aims to bridge the gap between the user and traditional physical hardware devices. Given the high learning curve in understanding how to use foreign technologies, we hope to break away from conventional control mechanisms and explore an intuitive way to control these devices. GestureIt will provide a tangible interface that relies on hand gestures to wirelessly control any device or software. By removing the distance between the user and traditional hardware devices our goal is to make GestureIt feel more like an extension of the body as opposed to an external machine.

1.3 Expected Outcome

The goal of this project is to capture simple hand gestures from the glove and use that input to wirelessly control a modified RC car. Controlled variables include speed and steering using a combination of flex, force and gyroscopic sensors. Multiple variables are controlled simultaneously as glove outputs a constant control signal.

The secondary goal of this project is to reach a level of comfort and precision with currently held by present day controllers. This is an important goal to achieve because if our glove cannot perform at least as well as current controllers, it will be ignored as people will favor the more efficient option.

1.4 Existing System/Application

Relatively recent developments in the field of glove-based input include the BarrettHand [1], a wired glove that can control hardware devices, which is a key feature in our design, though we improved upon it with wireless control to provide free range of motion. The Peregrine Gaming

Glove [2] provides software calibration, but has no support for macros and no support for multiple simultaneous key presses, which our prototype handles without trouble. The Reusch “Sonic Control” Glove [3] includes nice on-glove displays of volume and bass/treble levels, but is limited to controlling only the iPod, requires the user to press buttons on the glove, and does not sense gestures or pinches. We built a glove whose inputs are general enough to be used to control various devices and applications. Perhaps the model closest to our vision is the PowerGlove Remake [4, 5], which has wireless control, open Arduino architecture, and the ability to map input to multiple devices. Unfortunately, even this notable glove has its limitations and still requires deliberate button presses for its input, which is not the case for our glove. The most recent development is the G-Speak glove input system [6] that provides impressive gesture control, but also does not map to a hardware device and requires external sensors, limiting the range in which the user can roam and still provide input to the system. Our system can be used without external sensors and only requires the glove to be within the range of wireless communication with the hardware device. The G-Speak system seems to be highly intuitive though, an aspect that we strived to include and improve upon in our system by using gestures that do not require line of sight with the slave device but instead are simple and detached enough to be done behind your back. This control scheme is similar to that of the gyroscopic Loop Pointer [7] from Hillcrest Labs, which can be controlled from any hand position without pointing at the screen.

1.5 Software Requirement Specification

Table of Contents

1. Introduction

1.1. Purpose

1.2. Scope

1.3. Definition

1.4. References

2. Overall Description

2.1. Technologies to be used

2.2. Product Function

2.3. Constrains

1.1 Purpose

With gesture based hardware control system we aim to bridge the gap between humans and the systems they interact with, by allowing the user to control a hardware device using natural gestures. The goal of this project is to capture simple hand gestures from the Glove and use that input to wirelessly control a modified RC car. The purpose of undertaking this project is to understand the embedded systems and based upon that develop a Remote Controlled Car which can be controlled using Gestures of our Hands.

1.2 Scope

Outcome: Our goal is to understand the communication protocols and implement it to control our quad-copter by a smartphone (via Bluetooth).

Tasks: This project is divided into 6 different phases:

Phase 1: Assembly of Hardware (Sensors, motors, Arduino board)

Phase 2: Working with each component individually.

Phase 3: Integrating all Sensors together.

Material	Quantity	Cost (INR)
Arduino Due Microcontroller	1	1500
Arduino Pro Mini	1	500
Xbee Wireless Module	2	$1000 * 2 = 2000$
Flex Sensor	1	800
Force Sensor	2	$500 * 2 = 1000$
Accelerometer	1	300
Xbee USB Explorer	1	600
Xbee Shield	1	550
L393d H-Bridge Motor Driver	1	80
Total		7,330 INR

Phase 4: Wireless Connectivity of RC Car and Glove.

Phase 5: Integrating all hardware in a RC Car and Glove.

Phase 6: Final Test.

Costs: This project involves Hardware related costs:

1.3 Definition

Purpose of GestureIt is to bridge the gap between humans and the systems they interact with. To achieve this goal our group has designed and build a Gesture Based Hardware Control System which will be controlled through gestures of our hands. Many Input methods have been devised: keyboards, mice, joysticks, game controllers, and touch screens. But unfortunately, none of these devices remove the barrier between man and machine. With the Gesture Based Hardware Control system, we aim to remove this obstruction by allowing the user to control a hardware device using natural gestures. The Gesture Based Hardware Control takes advantage of a multitude of sensors to capture hand movements and uses this information to control a device – in this case, a modified Remote Controlled car.

2. Overall Descriptions

The GestureIt is a gesture controller that can be easily interfaced with hardware or software via a wireless connection. The glove implements an array of sensors plus filtering circuitry and

mathematical analysis firmware to derive state values for each sensor and transmits them to an authorized receiver. We have used the glove to drive an RC car.

2.1 Technologies to be used

- C Language for Programming Arduino Microcontroller.
- Arduino IDE.
- X-CTU IDE for Configuring Xbee Wireless Modules.

2.2 Product Functions

- Send/receive signals from Arduino Due to Arduino Pro Mini via Xbee.
- Calibrate flight controller
- Interpret commands from Sensors on glove and send them to Arduino for Processing.

Commands:

- Variable Acceleration using Flex Sensor.
- Drive Forward/backward using Different Force Sensors.
- Turn Left/Right using Accelerometer.

2.3 Constraints

- Car operate distance not more than 50m in eye sight from the wireless controller.
- Car can only run for 20 minutes.

CHAPTER 2

2. METHODOLOGY

Our prototype can be described in three parts: the glove, the controller, and the slave device (an RC car in this scenario).

Control of vehicle motion is achieved by altering resistance of sensors on our glove. These Signals are sent to controller unit for processing, from the control unit it goes to the slave device for execution of commands.

2.1 Product / system view

2.1.1 The Glove

The glove is fitted with a Flex sensor, 2 Force sensors and an Accelerometer. Each sensor is carefully sewn into the glove using double stitching to ensure durability. Flex Sensor is on Index finger, Force sensor 1 is on Ring Finger and Force sensor 2 is on Pinky Finger, Accelerometer is sewn to the wrist, adjacent to the protoboard which accepts connections from all the sensors and the cable from the control unit.

2.1.2 The Control Unit

The control unit contains an Arduino Due, a prototyping shield, XBee wireless unit, and a 9V battery. The components are housed in a box. The control unit's prototyping shield accepts connections to and from the glove and holds voltage divider circuits for the sensors.

2.2.3 Modified RC Car

The car is an RC car with its inner circuitry gutted and replaced with an Arduino mini, H-bridge controlled motors, Servo Motor, DC Motor. An H-Bridge allows the Arduino to control the acceleration and turning of DC motors which were already mounted in the chassis. The Arduino regulates this to 5V for use with the H-Bridge, and a separate 3.3V regulator is used to supply power to the XBee.

2.2 System components & Functionalities

2.2.1 Arduino Due:

The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. It is the first Arduino board based on a 32-bit ARM core microcontroller. It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTs (hardware serial ports), a 84 MHz clock, an USB OTG capable connection, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button.

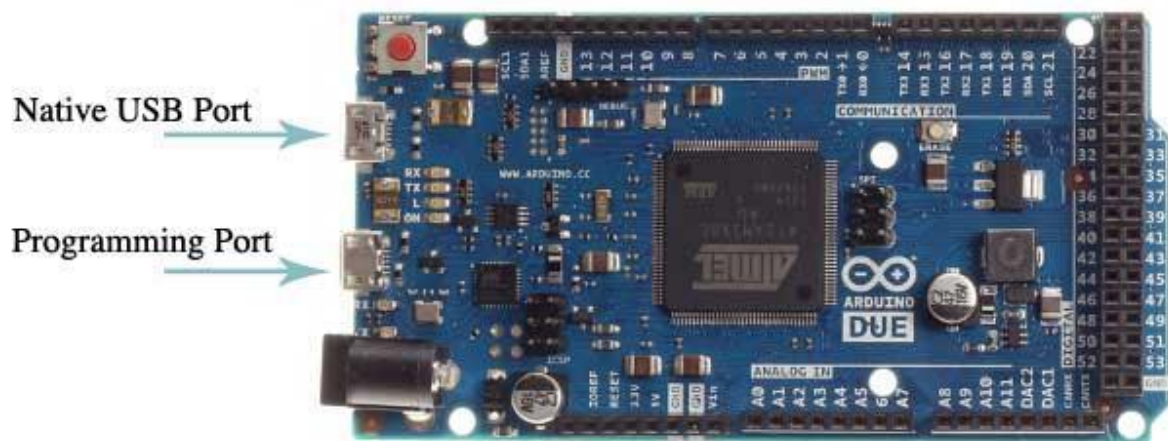


Fig 2.1: Arduino DUE Microcontroller

The board contains everything needed to support the microcontroller; simply connect it to a computer with a micro-USB cable or power it with an AC-to-DC adapter or battery to get started [8]. The Due is compatible with all Arduino shields that work at 3.3V and are compliant with the 1.0 Arduino pinout.

The Due follows the 1.0 pinout:

TWI: SDA and SCL pins that are near to the AREF pin.

IOREF: allows an attached shield with the proper configuration to adapt to the voltage provided by the board. This enables shield compatibility with a 3.3V board like the due and AVR-based boards which operate at 5V.

An unconnected pin, reserved for future use.

Power:

The Arduino Due can be powered via the USB connector 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.

Memory:

The SAM3X has 512 KB (2 blocks of 256 KB) of flash memory for storing code. The bootloader is pre burned in factory from Atmel and is stored in a dedicated ROM memory. The available

SRAM is 96 KB in two contiguous bank of 64 KB and 32KB. All the available memory (Flash, RAM and ROM) can be accessed directly as a flat addressing space.

It is possible to erase the Flash memory of the SAM3X with the onboard erase button. This will remove the currently loaded sketch from the MCU. To erase, press and hold the Erase button for a few seconds while the board is powered.

AVR Arduino microcontroller Specifications

Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz
Length	101.52 mm
Width	53.3 mm
Weight	36 g

2.2.2 Arduino Mini Pro

The Arduino Pro Mini is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, an on-board resonator, a reset button, and holes for mounting pin headers. A six pin header can be connected to an FTDI cable or breakout board to provide USB power and communication to the board.

The Arduino Pro Mini is intended for semi-permanent installation in objects or exhibitions. The board comes without pre-mounted headers, allowing the use of various types of connectors or direct soldering of wires. The pin layout is compatible with the Arduino Mini.

There are two version of the Pro Mini. One runs at 3.3V and 8 MHz, the other at 5V and 16 MHz.

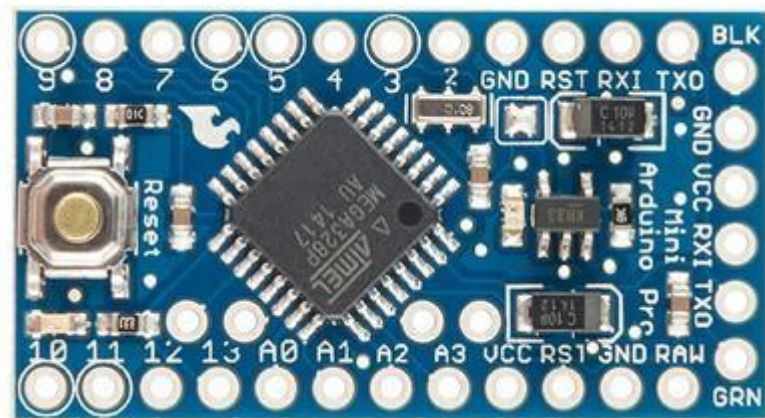


Fig 2.2: Arduino pro mini Microcontroller

Power:

The Arduino Pro Mini can be powered with an FTDI cable or breakout board connected to its six pin header, or with a regulated 3.3V or 5V supply (depending on the model) on the Vcc pin. There is a voltage regulator on board so it can accept voltage up to 12VDC. If you're supplying unregulated power to the board, be sure to connect to the "RAW" pin on not VCC.

The power pins are as follows:

RAW. For supplying a raw voltage to the board.

VCC. The regulated 3.3 or 5 volt supply.

GND. Ground pins

Memory:

The ATmega328 has 32 kB of flash memory for storing code (of which 0.5kB is used for the bootloader). It has 2 kB of SRAM and 1kB of EEPROM (which can be read and written with the EEPROM library).

2.2.3. Flex Sensor

A simple flex sensor 4.5" in length. As the sensor is flexed, the resistance across the sensor increases. Patented technology by Spectra Symbol - they claim these sensors were used in the original Nintendo Power Glove [9].

The resistance of the flex sensor changes when the metal pads are on the outside of the bend (text on inside of bend).

Connector is 0.1" spaced and bread board friendly.

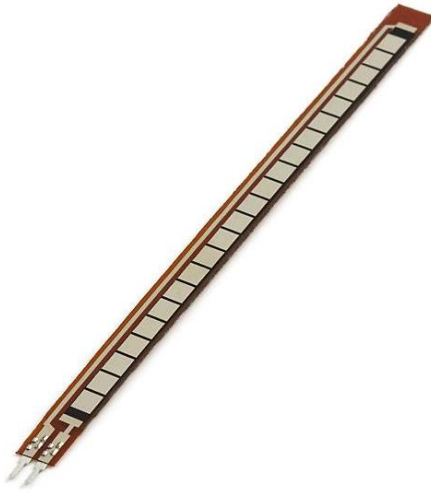


Fig 2.3: Flex Sensor

Features

- Angle Displacement Measurement
- Bends and Flexes physically with motion device
- Mechanical Specifications
- Life Cycle: >1 million
- Height: 0.43mm (0.017")
- Temperature Range: -35°C to +80°C
- Electrical Specifications
- Flat Resistance: 10K Ohms
- Resistance Tolerance: $\pm 30\%$
- Bend Resistance Range: 60K to 110K Ohms
- Power Rating: 0.50 Watts continuous. 1 Watt Peak

2.2.4. Force Sensor



Fig 2.4: Force Sensor

This is a small force sensitive resistor. It has a 0.16" (4 mm) diameter active sensing area. This FSR from Interlink Electronics will vary its resistance depending on how much pressure is being applied to the sensing area. The harder the force, the lower the resistance. When no pressure is

being applied to the FSR, its resistance will be larger than $1\text{M}\Omega$, with full pressure applied the resistance will be $2.5\text{k}\Omega$.

Two pins extend from the bottom of the sensor with 0.1" pitch making it breadboard friendly.

These sensors are simple to set up and great for sensing pressure, but they aren't incredibly accurate. Use them to sense if it's being squeezed, but you may not want to use it as a scale.

Dimensions:

- Overall length: 1.75"
- Overall width: 0.28"
- Sensing area: 0.3"

Features:

- Actuation Force as low as 2 grams
- Wide force sensitivity range 0.1N - 10N

2.2.5. Triple Axis Accelerometer Breakout - ADXL335:

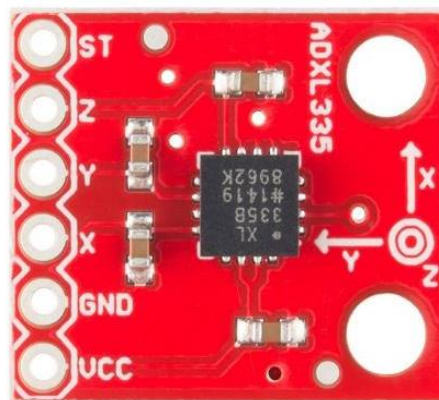


Fig 2.5: Accelerometer

Breakout board for the 3 axis ADXL335 from Analog Devices. This is the latest in a long, proven line of analog sensors - the holy grail of accelerometers. The ADXL335 is a triple axis MEMS accelerometer with extremely low noise and power consumption - only 320uA! The sensor has a full sensing range of $\pm 3g$.

There is no on-board regulation, provided power should be between 1.8 and 3.6VDC [10].

Board comes fully assembled and tested with external components installed. The included 0.1uF capacitors set the bandwidth of each axis to 50Hz.

Dimensions: 0.7x0.7"

2.2.6. L293D Dual H- Bridge Motor Driver IC



Fig 2.6: L293D IC

L293D is quadruple high-current half-H driver. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive supply applications. All inputs are TTL compatible.

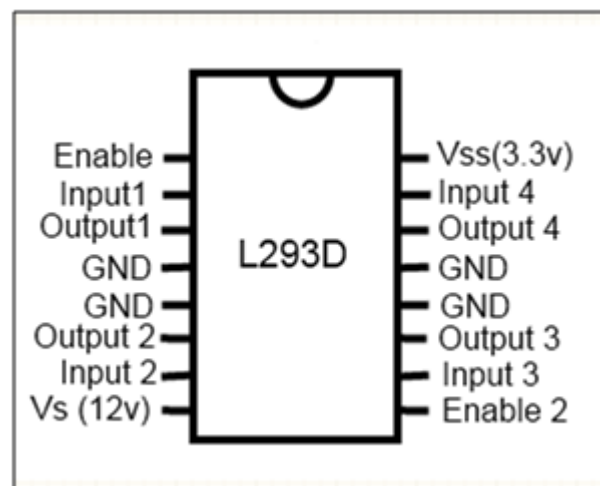


Fig 2.7: L293D IC Pin Diagram

Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1, 2 EN and drivers 3 and 4 enabled by 3, 4 EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full - H (or bridge) reversible drive suitable for solenoid or motor applications.

2.2.7. XBee Wireless Communication Module Wire Antenna



Fig 2.8: Xbee Module

This is the very popular 2.4GHz XBee module from Digi. These modules take the 802.15.4 stack (the basis for Xbee) and wrap it into a simple to use serial command set. These modules allow a very reliable and simple communication between microcontrollers, computers, systems, really anything with a serial port! Point to point and multipoint networks are supported. This XBee wireless device can be directly connected to the serial port (at 3.3 V level) of your microcontroller. By using a logic level translator it can also be interfaced to 5V logic (TTL) devices having serial interface. This module supports data rates of up to 115 kbps. 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.

2.2.8. DC Motor



Fig 2.9: DC Motor

These are standard '130 size' DC hobby motors. They come with a wider operating range than most toy motors: from 4.5 to 9VDC instead of 1.5-4.5V. This range makes them perfect for controlling with an Adafruit Motor Shield, or with an Arduino where you are more likely to have 5 or 9V available than a high current 3V setting. They'll fit in most electronics that already have 130-size motors installed and there's two breadboard-friendly wires soldered on already for fast prototyping

Features:

- Operating Temperature: -10°C ~ +60°C
- Rated Voltage: 6.0VDC

- Rated Load: 10 g*cm
- No-load Current: 70 mA max
- No-load Speed: 9100 \pm 1800 rpm
- Loaded Current: 250 mA max
- Loaded Speed: 4500 \pm 1500 rpm
- Starting Torque: 20 g*cm
- Starting Voltage: 2.0
- Stall Current: 500mA max
- Body Size: 27.5mm x 20mm x 15mm
- Shaft Size: 8mm x 2mm diameter
- Weight: 17.5 grams

2.2.9. Xbee Shield



Fig 2.10: Xbee Shield

The Xbee shield allows an Arduino board to communicate wirelessly using Xbee. It is based on the Xbee module from MaxStream. The module can communicate up to 100 feet indoors or 300 feet outdoors (with line-of-sight). It can be used as a serial/usb replacement or you can put it into a command mode and configure it for a variety of broadcast and mesh networking options. The shields breaks out each of the Xbee's pins to a through-hole solder pad. It also provides female pin headers for use of digital pins 2 to 7 and the analog inputs, which are covered by the shield (digital pins 8 to 13 are not obstructed by the shield, so you can use the headers on the board itself).

The Arduino XBee shield can be used with different XBee modules. The instructions below are for the XBee 802.15.4 modules (sometimes called "Series 1" to distinguish them from the Series 2 modules, although "Series 1" doesn't appear in the official name or product description).

Features:

- Mounts directly onto your Arduino
- Can be use as XBEE development board
- Selectable level Shifting (5V & 3V3)
- 3.3V power regulation and level shifting on-board
- 12x11 grid of 0.1" spaced prototyping holes
- All XBEE pins are available at male Headers
- Reset button brought out
- Power, DIN, DOUT, RSSI, Sleep and DIO5 indicator LEDs

2.2.10. Xbee USB Adapter

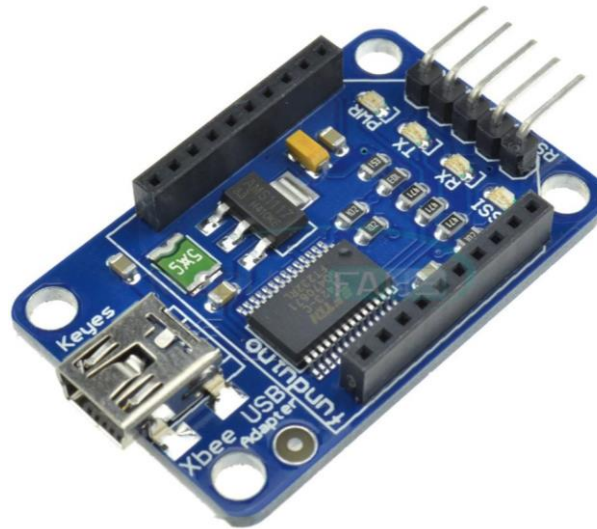


Fig 2.11: Xbee USB Adapter

Xbee USB adapter is designed for the XBee module configuration parameters to facilitate using or to work as a FTDI writer. The new version of Xbee USB adapter is also welded with the female pin for FTDI so that users can directly use the adapter as a FTDI program writer, such as Arduino FIO/pro/mini Arduino and so on. It's a must-have for interactive electronic installations and projects.

The adapter allows the PC to configure XBee / Bluetooth module and the controller / development board for wireless data transmission via mini USB cable. You can upload your program through the wireless controller module, or conduct wireless real-time data transmission. The adapter uses a FTDI232 USB-UART converter chip to ensure stable and reliable data transmission.

Features:

- Used for the XBee module configuration parameters
- Easily connected to a PC via mini USB cable
- XBee-setting support software X-CTU
- Also used as a USB-TTL adapter
- Voltage: +5V(USB Power)

CHAPTER 3

3. DESIGN CRITERIA

3.1 System Design

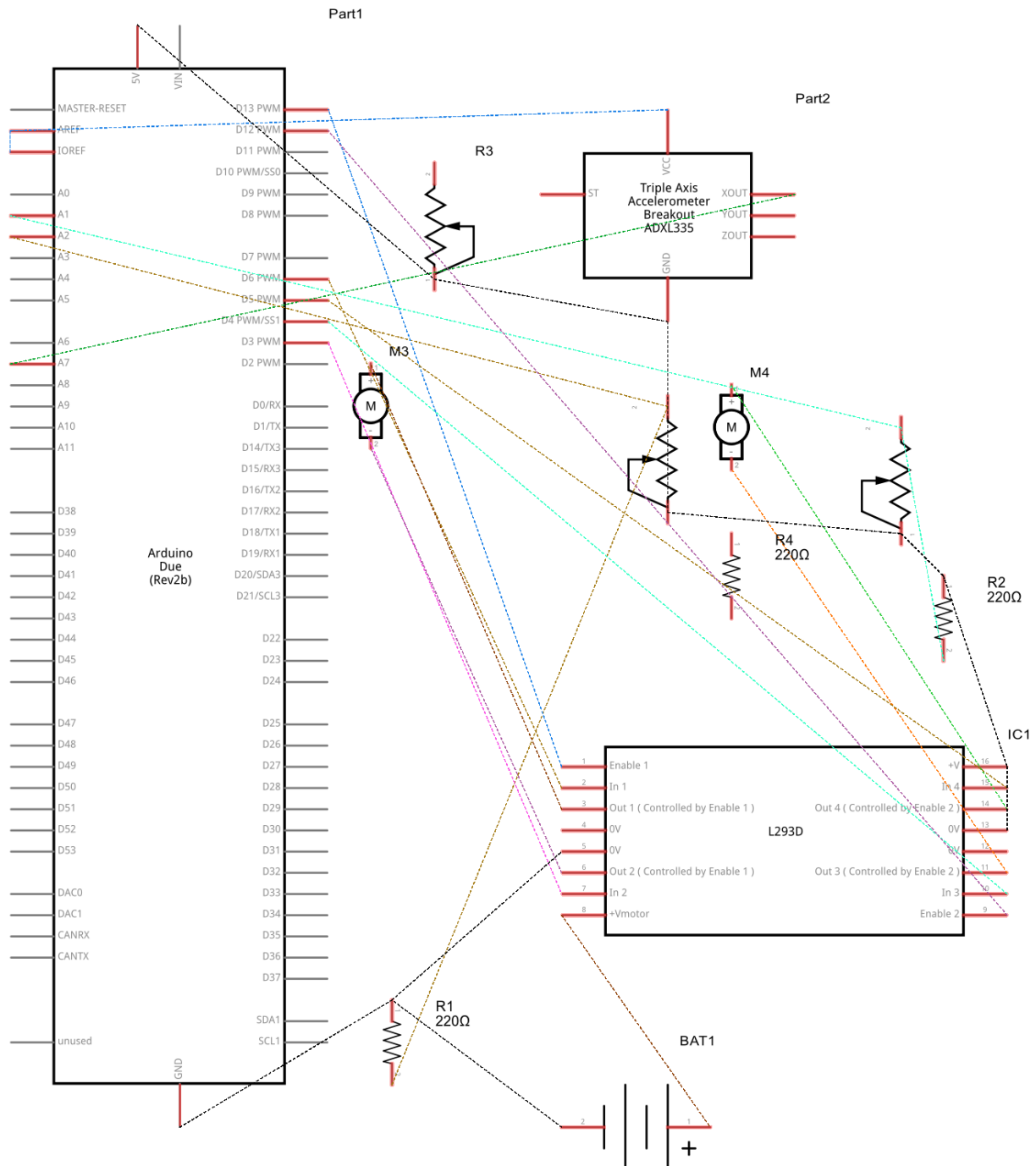


Fig 3.1: Schematic diagram of the Main Circuit

3.2 Design Diagram

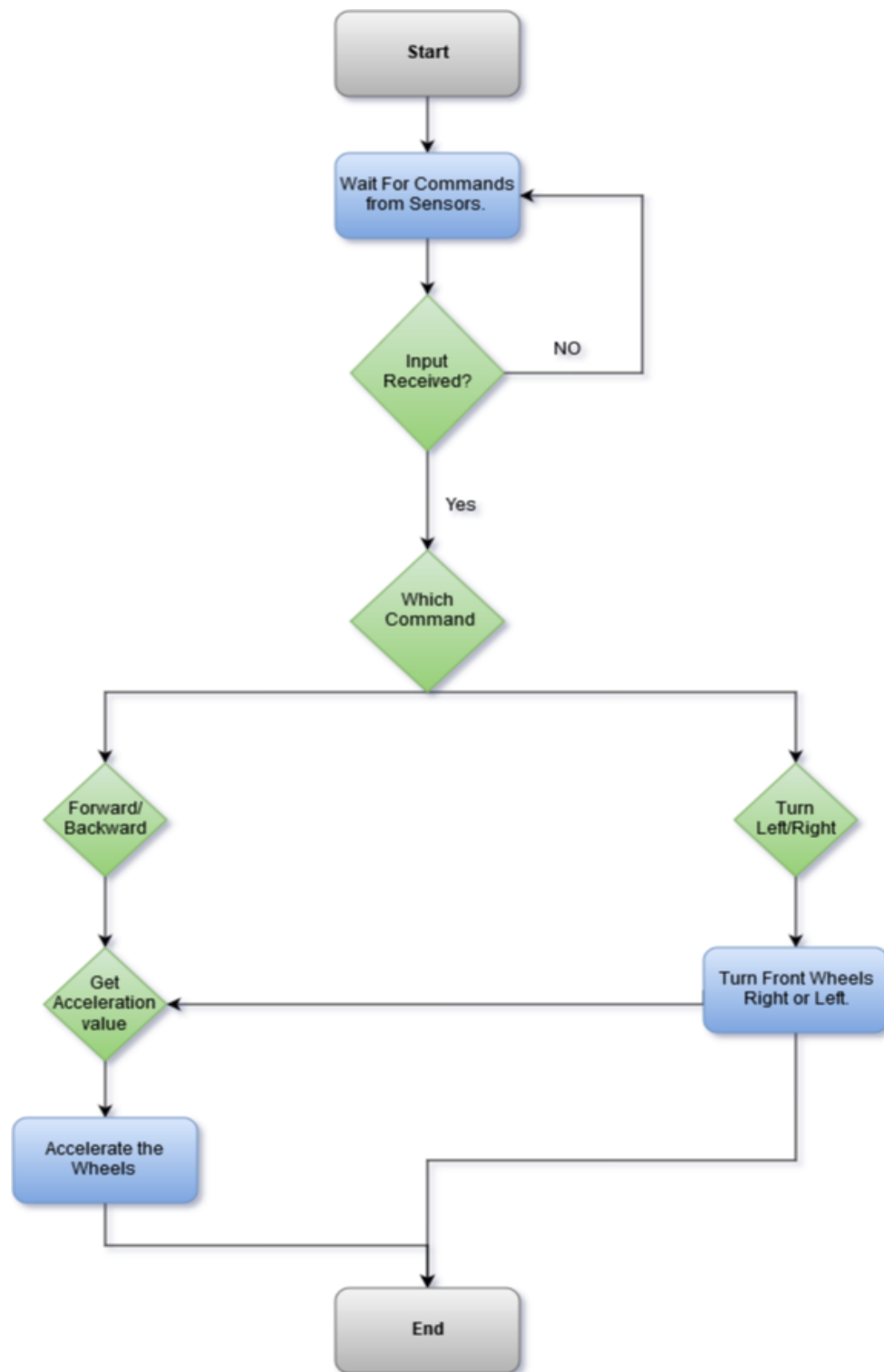


Fig 3.2: Flowchart of GestureIt

3.3 System Architecture

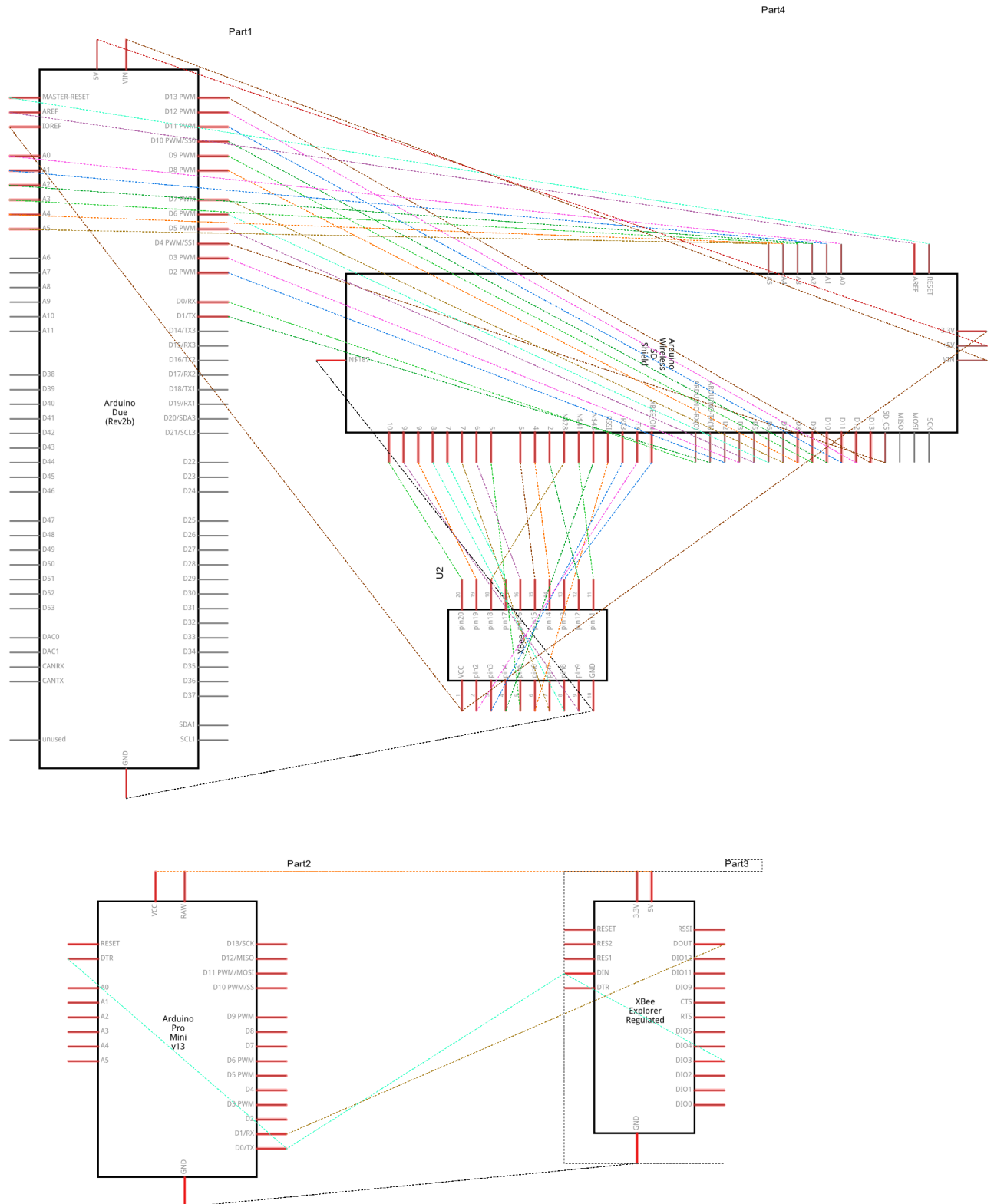


Fig 3.3: System Architecture for GestureIt

CHAPTER 4

4. DEVELOPMENT & IMPLEMENTATION

4.1 System Modules and flow of implementations

4.1.1 Motion Capture

The glove itself is nothing more than a neoprene glove bought at a sporting goods store. When fitted with all of its sensors, a sophisticated hand-gesture capture device is created. The flex sensor is sewn into the back of the index finger because the index finger is easily bent independent of other fingers, the force sensors are sewn to the tips of the ring and pinky fingers because they are easily pressed with the thumb. The 2D gyroscope is sewn into the wrist because that is the part of the hand that rotates. With the combined data of these sensors, a fairly detailed motion can be extrapolated. The performance of the motion capture is rated based on ease of use, the natural feeling of the necessary motions, and the recorded sensor readings. The sensor readings were mapped to cover the entire range of motion values to ensure the highest resolution possible.

4.1.2 Wireless Communication

The Arduino mega communicates wirelessly with the Arduino mini via two XBee modules on the same Personal Area Network (PAN). The performance of the wireless communication was evaluated based on how well the car would respond to wireless commands versus how it responded to commands sent directly over USB. The Arduino mini sends out a specific char 'a' when it is ready to start receiving data from the Arduino mega; the Arduino mega reads this char and sends out a packet of numbers preceded by another specific char '.'. When the Arduino mini reads a period, it knows that the Arduino mega sent it a packet of numbers and reads it [11]. After reading the packet and doing whatever the data told it to do, the Arduino mini sends out another char 'a'. Once the special characters were used, wireless communication performed as well as the USB communication and was deemed successful.

4.1.3 Controlling the Motors

To start with, each motor is controlled by a digital output Arduino Mini Pro. It sends a High or Low signal to motor for its operations. Variable acceleration is controlled through the values coming from Flex sensor. Values coming from Flex sensor (0-1023) is mapped between the operational values of DC Motor (0-255).

Rotation of motor in clockwise direction or anti-clockwise direction is controlled by the values coming from Force Sensors [12]. We are using two Force sensors using which we can change the directions of motor rotation. If Force sensor 1 is set to High then DC will rotate Clockwise direction and Force sensor 2 will be set to Low, Else if Force Sensor 2 is set to High then DC will rotate anti-clockwise direction and Force sensor 1 is set to Low.

```
// Run the motor to clockwise direction

void loop() {
  analogWrite(3,255);
  analogWrite(6,0);
}
```

```

// Run the motor to anti-clockwise direction

void loop() {
  analogWrite(3,0);
  analogWrite(6,255);
}

// Variable Acceleration using the values coming from Flex sensor.

int flexposition; // Input value from the analog pin.
int dcposition; // Output value to the servo.
flexposition = analogRead(flexpin);
dcposition = map(flexposition, 782, 950,255, 0);
dcposition = constrain(dcposition, 0, 255);

// Integrating Flex and DC
/*****
* Controlling DC motor using Flex Sensor.
* Rishabh Verma, Rishi Chandak, Saurabh Saluja
* Computer Science and Engineering Department
* Sharda University, Batch 2012-2016
* *****/

const int flexpin = 0;
void setup() {
  Serial.begin(9600);
  pinMode(13,OUTPUT);
  digitalWrite(13,HIGH);
}

void loop() {

  int flexposition; // Input value from the analog pin.
  int dcposition; // Output value to the servo.
  flexposition = analogRead(flexpin);
  dcposition = map(flexposition, 782, 950,255, 0);
  dcposition = constrain(dcposition, 0, 255);
  analogWrite(6,dcposition);
  Serial.print("sensor: ");
  Serial.print(flexposition);
  Serial.print(" dc: ");
  Serial.println(dcposition);
  delay(150);
}

// Integrating Force, Flex and DC
/*****
* Integrating Force, Flex and DC Motor.
* Rishabh Verma, Rishi Chandak, Saurabh Saluja
* Computer Science and Engineering Department
* Sharda University, Batch 2012-2016
* *****/
const int forcePin1 = 2;
const int forcePin2 = 1 ;

```

```

const int flexPin = 0;

void setup() {
  Serial.begin(9600);
  pinMode(13,OUTPUT);
  digitalWrite(13,HIGH);
}

void loop() {
  int forcePosition1; // Input value from the analog pin.
  int forcePosition2;
  int flexPosition;// Input value from the analog pin.
  int dcposition; // Output value to the dc.
  forcePosition1 = analogRead(forcePin1);
  delay(10);
  forcePosition2 = analogRead(forcePin2);
  delay(10);
  flexPosition = analogRead(flexPin);
  delay(10);
  Serial.println(flexPosition);

  /* Force Mapping */

  if(forcePosition1 <800 && forcePosition2 < 800){
    Serial.println("Error : Multiple Sensor Input Received on single Motor");
    dcposition = 255;

  /* Force Mapping */

    dcposition = constrain(dcposition, 0, 255);
    analogWrite(3,dcposition);
    analogWrite(6,dcposition);
  }
  else if(forcePosition1<800)
  {
    Serial.println("Force Sensor 1 Activated");
    dcposition = map(flexPosition, 800, 1023,255, 0);

  /* Force Mapping */

    dcposition = constrain(dcposition, 0, 255);
    analogWrite(6,dcposition);
  }
  else if(forcePosition2<800)
  {
    Serial.println("Force Sensor 2 Activated");
    dcposition = map(flexPosition, 800, 1023,255, 0);

  /* Force Mapping */

    dcposition = constrain(dcposition, 0, 255);
    analogWrite(3,dcposition);
  }
}

```

```

    else{
        Serial.println("No Directional Input Received");
        dcposition = 255;

        /* Force Mapping */

        dcposition = constrain(dcposition, 0, 255);
        analogWrite(3,dcposition);
        analogWrite(6,dcposition);
    }
    delay(150);
}

```

4.1.4 Putting It All Together

```

/*****
* Controlling Car using Flex,Force and Accelerometer.
* Rishabh Verma, Rishi Chandak, Saurabh Saluja
* Computer Science and Engineering Department
* Sharda University, Batch 2012-2016
* *****/

//Define pins for Communication

const int forcePin1 = 2;
const int forcePin2 = 1 ;
const int flexPin = 0;
const int xPin=A3;
int sensorvalue =0; //Acceleration Sensor
int forcePosition1; // Input value from the analog pin.
int forcePosition2;
int flexPosition;// Input value from the analog pin.
int dc1position; // Output value to the dc.
int dc2position; // Output value to the dc.

void setup() {
    Serial.begin(9600);
    pinMode(A3,INPUT);// voltage divider value input
    pinMode(13,OUTPUT);
    digitalWrite(13,HIGH);
    pinMode(12,OUTPUT);
    digitalWrite(12,HIGH);
}

void loop() {
    forcePosition1 = analogRead(forcePin1);
    delay(10);
    forcePosition2 = analogRead(forcePin2);
    delay(10);

```

```

flexPosition = analogRead(flexPin);
delay(10);
Serial.println(flexPosition);
sensorvalue = analogRead(xPin); //read analog value from sensor

```

```

/*Turning The Forward Wheels*/

```

```

if(sensorvalue>560) //Value from Accelerometer
{
    dc2position=0;
    dc2position = constrain(dc2position, 0, 255);
    analogWrite(7,dc2position);
}
else if(sensorvalue<460)//Value from Accelerometer
{
    dc2position=0;
    dc2position = constrain(dc2position, 0, 255);
    analogWrite(4,dc2position);
}
else //Keeping Axel at 90 Degree [Straight]
{
    dc2position=255;
    dc2position = constrain(dc2position, 0, 255);
    analogWrite(7,dc2position);
    analogWrite(4,dc2position);
}
Serial.print("DC Motor 2 Value : ");
Serial.print(dc2position);

```

```

/* Force Mapping */

```

```

if(forcePosition1 <800 && forcePosition2 < 800){
    Serial.println("Error : Multiple Sensor Value on single Motor.");
    dc1position = 255;

    /* Force Mapping */

    dc1position = constrain(dc1position, 0, 255);
    analogWrite(3,dc1position);
    analogWrite(6,dc1position);
}
else if(forcePosition1<800)
{
    Serial.println("Force Sensor 1 Active");
    dc1position = map(flexPosition, 800, 1023,255, 0);

```

```

/* Force Mapping */

dc1position = constrain(dc1position, 0, 255);
analogWrite(6,dc1position);
}
else if(forcePosition2<800)
{
  Serial.println("Force Sensor 2 Active");
  dc1position = map(flexPosition, 800, 1023,255, 0);

/* Force Mapping */

  dc1position = constrain(dc1position, 0, 255);
  analogWrite(3,dc1position);
}
else{
  Serial.println("Waiting For Input From Sensor");
  dc1position = 255;

/* Force Mapping */
  dc1position = constrain(dc1position, 0, 255);
  analogWrite(3,dc1position);
  analogWrite(6,dc1position);
}
delay(150);
}

```

4.2 Critical Module

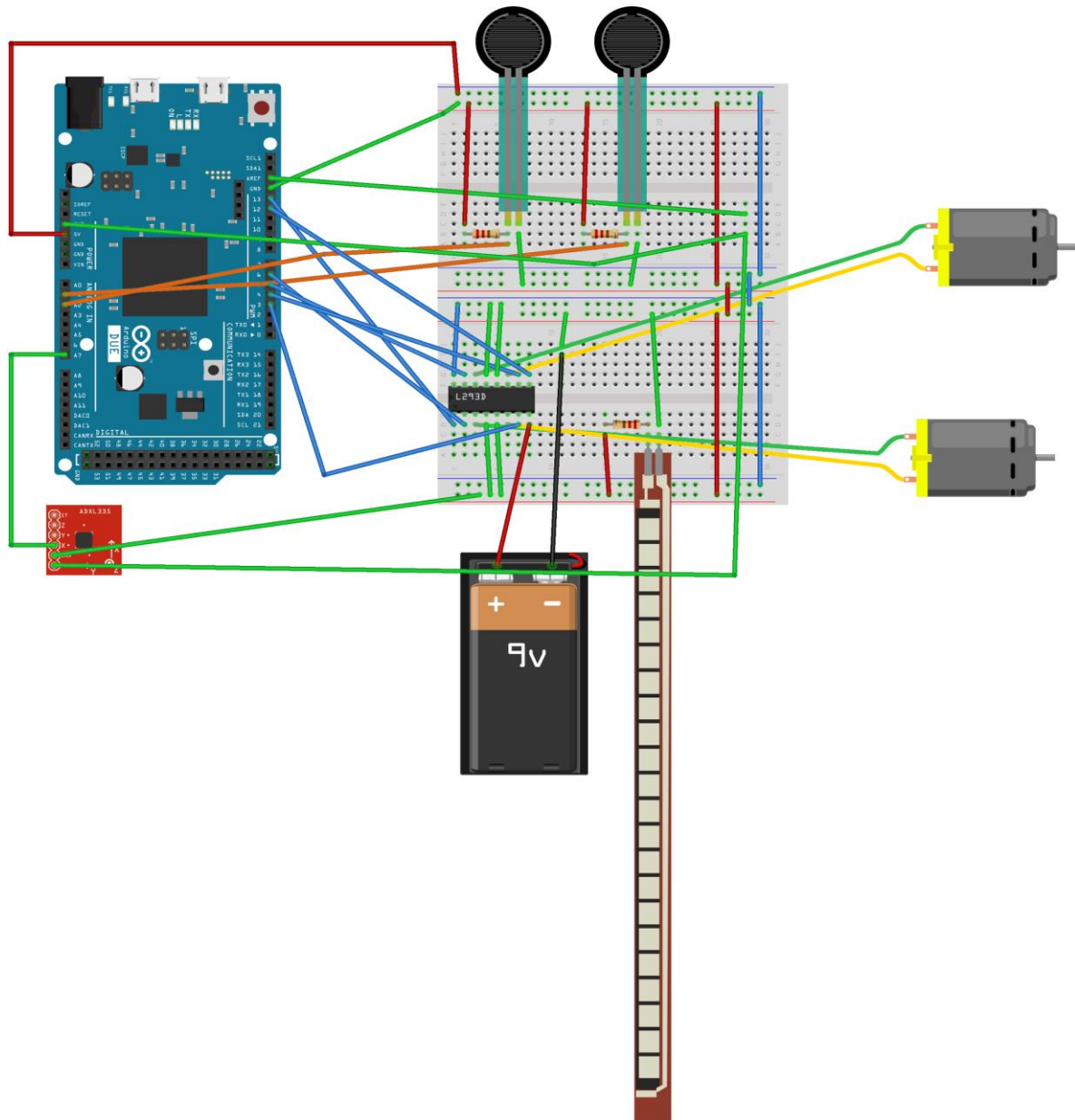


Fig 4.1: Circuit Diagram for GestureIt

CHAPTER 5

5. RESULTS & TESTING

5.1. Result

5.1.1. Success Cases

Firstly we were unable to get proper inputs from our 2D Gyroscope, Inputs that we captured were too noisy (as too many inputs were coming from gyroscope) for turning the car properly so we replaced the gyroscope with Accelerometer and captured the values of X-Axis only for turning.

We were stuck in wireless transmission of data from Xbee mounted on Arduino DUE to the one mounted on Arduino Pro Mini. We solved this by configuring one Xbee as Coordinator AT and the other one as Router AT. Previously we configured it as Coordinator AT and End Device AT.

5.1.2. Failure Cases

Firstly we were unable to turn the car in appropriate direction as too many values were coming from our 2D gyroscope and we were unable to buffer the required data and use the required signals only. As a result of which our turning was not smooth.

Secondly we were having problems with transmitting data wirelessly. We paired both the Xbee modules but they were not communicating as we expected. The data received on the other end was not usable as it was appended with some other foreign data as well and we were receiving some random special characters on serial monitor.

5.2. Testing

5.2.1. Type of testing adapted

5.2.1.1. Unit Testing

Each module was developed individually and tested separately before integrating with other modules. First we made circuit for Flex Sensor and DC motor for variable acceleration. After this we worked on Force Sensor and Motors then with Accelerometer and finally with Xbee Wireless Modules.

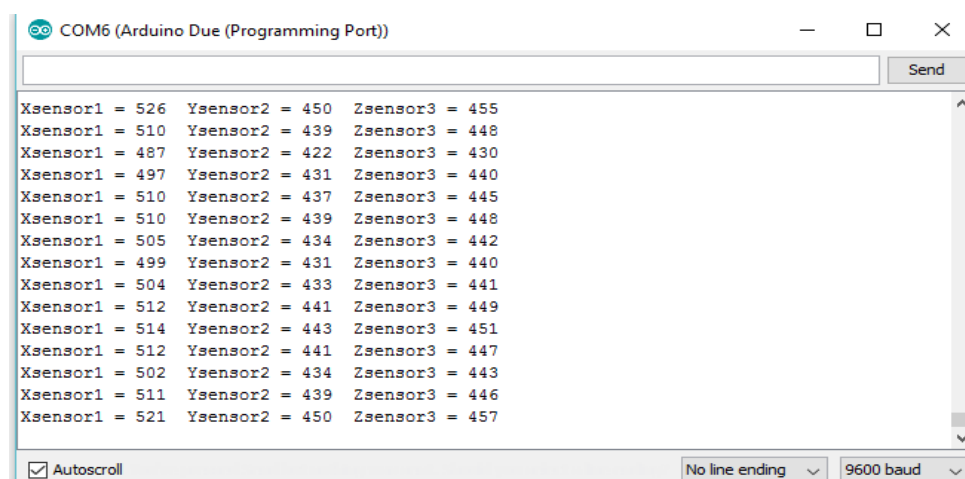
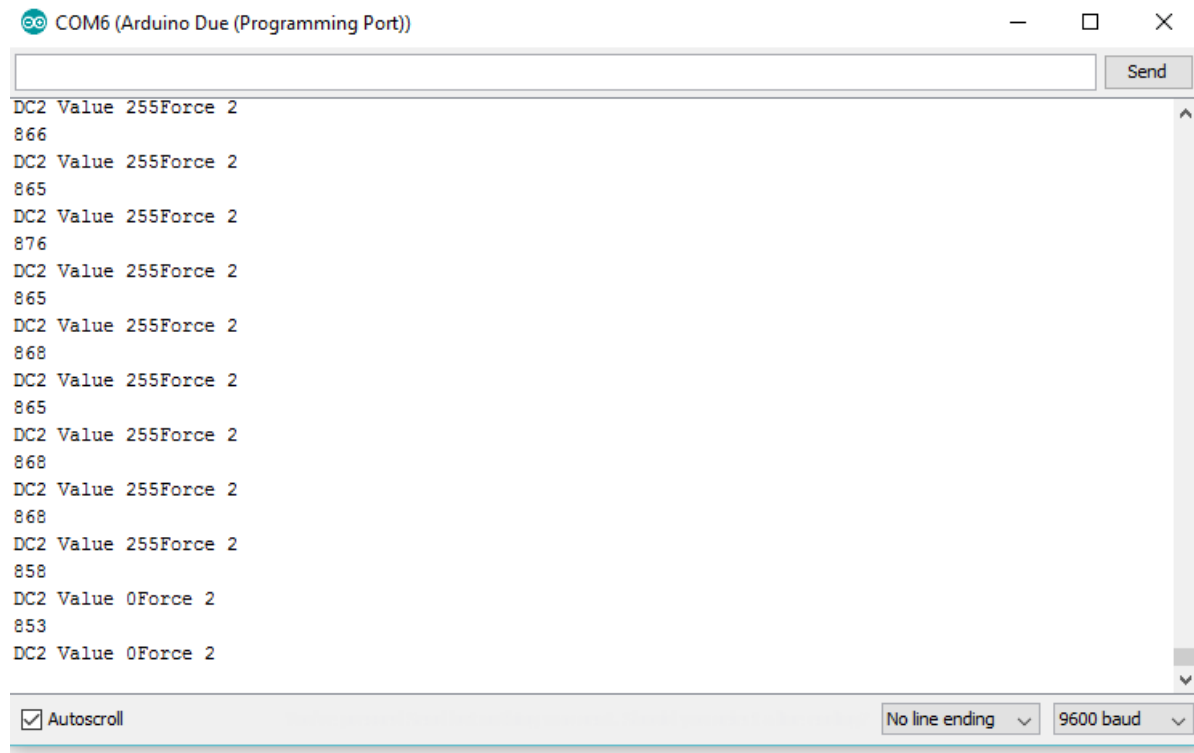


Fig 5.1: Screen Monitor Output 1

5.2.1.2. Integration testing

After we finished developing circuits for individual components and testing every module, we assembled and integrated all the modules together and tested how each and every module functions when integrated with other modules.



The screenshot shows the Arduino IDE Serial Monitor window for COM6 (Arduino Due (Programming Port)). The window displays a series of data points in a text-based format. The data is organized into three groups, each starting with a header line: 'DC2 Value 255Force 2', 'DC2 Value 255Force 2', and 'DC2 Value 0Force 2'. Each header is followed by a list of numerical values. The first two groups show values ranging from 853 to 876, while the third group shows values 853 and 858. The window includes a 'Send' button at the top right, a scroll bar on the right, and a status bar at the bottom with 'Autoscroll' checked, 'No line ending' selected, and '9600 baud' selected.

```

COM6 (Arduino Due (Programming Port))
DC2 Value 255Force 2
866
DC2 Value 255Force 2
865
DC2 Value 255Force 2
876
DC2 Value 255Force 2
865
DC2 Value 255Force 2
868
DC2 Value 255Force 2
865
DC2 Value 255Force 2
868
DC2 Value 255Force 2
868
DC2 Value 255Force 2
858
DC2 Value 0Force 2
853
DC2 Value 0Force 2

```

Fig 5.2: Screen Monitor Output 2

5.2.1.3. System Testing

The final test was to drive the car. We sew all the sensors on a glove and soldered it joints with wires with goes to Arduino Due. All the components were working correctly as expected and drive duration was approximately 20 minutes.

5.2.2. Conclusion of Testing

The RC car is fully operable with each and every module being tested completely and integrated. Both the motors are operating as per the signals received. All sensors are also working together to capture hand gestures and transmit captured signals to Arduino.

CHAPTER 6

6. Conclusion & Future Improvements

6.1. Performance Estimation

6.1.1. Wireless Sensor

XBee is built on top of the IEEE 802.15.4 standard which defines the Medium Access Control (MAC) and physical layers, operating in an unlicensed band of 2.4 GHz with a data transfer rate of 250 kbps. In terms of networking capability, XBee protocol supports three types of communication topologies such as point-to-point, point-to-multipoint and mesh topology. XBee enabled devices operate with low duty cycle, hence providing longer battery life which makes it the most widely used devices for a wireless sensor network. XBee protocol also features multi-hop communication capability, therefore providing a vast range of communication and a wide coverage area.

XBee is ideal for smart applications, it also has few drawbacks. For instance, XBee supports low data rate, and this may make it inadequate for an industrial application which would have a higher requirement in terms of network bandwidth, reliability, scalability.

6.1.2. Energy Consumption and Battery Life Time Prediction

Energy efficiency is one of the important functional indexes since it directly affects the life cycle of the system. Our System is using two 9V battery for powering 2 DC Motors and Arduino Due and another 6 Alkaline AA battery for Arduino Mini with mounted Xbee on it for wireless communication. The 9V batteries last for 20 Minutes with continuous operations.

6.2. Usability of Product / system

At present, we are able to control multiple Motors by capturing gestures of our hands using multitude of sensors. By adding more sensors and powerful motors, the practical applications can be expanded.



Fig 6.1: Smart glove (Back)



Fig 6.2: Smart glove (Front)



Fig 6.3: Smart car

6.3 Limitations

There are few limitations that we discovered in our Project:

- The range of Xbee S2 is 90 meters max.
- With a new battery it can run for 20 Minutes. The first problem is how to reduce that 9V to 5V that the Arduino board can use. Most Arduino boards have an external voltage input, and a range of 7-12V is recommended. So 9V seems perfect. The problem is that most Arduino boards use a linear regulator to drop that 9V to 5V. If you are drawing a mere 50mA, 0.2W is being burnt in this linear regulator with 0.25W being used by the Arduino itself.
- 9V batteries are very low capacity, an alkaline 9V PP3 has a capacity of between 500 and 600mAh. This really isn't very high – a typical alkaline AA battery will be at least 2000mAh. 6 series AA cells will provide the same voltage, but a capacity at least 4 times the size!

6.4. Scope of Improvement

- As we all know that wireless medium is full of noises and data coming may get lost or received corrupted. Other devices that typically operate in the 2.4 GHz band and can cause interference include baby monitors, remote control cars, old wireless intercom. Different Libraries and API's could be used to make the wireless communication more effective.
- More sensors could be attached to system for making the device more responsive. For example using IR obstacle sensor for this RC car.
- By mounting more powerful batteries we could extend the execution process of our system.
- We can extend the range of wireless communication by installing Wi-Fi modules instead of Xbee modules.

6.5 Conclusion

The GestureIt system demonstrates the use of intuitive, simple, glove-based input to control the movement of external entities - a remote control car in this case. The flawless data transfer from the glove to the car justifies the effective and operable features of the glove based on how well the system controlled the RC car, we are convinced the GestureIt system is a viable user interface with untapped capabilities. Though only a prototype, further development would undoubtedly produce an even more accurate and intuitive design. The GestureIt system has the potential to start bridging the large gap between human ideas and machine responses.

References

1. Barrett Hand Glove Control. <http://www.youtube.com/watch?v=Ywo6WR9NUpE>.
2. Peregrine Gaming Glove. <http://www.engadget.com/2009/12/18/peregrine-gaming-glove-modeled-calibrated-and-demoed-on-video/>.
3. Sonic Control Glove. <http://fibretronic.com/news/Reusch%20Sonic%20Control%20Glove>.
4. Nintendo Power Glove http://en.wikipedia.org/wiki/Power_Glove.
5. [Michael McRoberts](#), Beginning Arduino 1st ed. 2010 Edition, Pg:125-130
6. 20th Anniversary Power Glove Remake. <http://www.instructables.com/id/Power-Glove-20th-Anniversary-Edition/>.
7. Oblong Gesture Computing http://www.readwriteweb.com/archives/minority_report_in_your_living_room_gestural_inter.php.
8. [Massimo Banzi](#), Getting Started with Arduino Second Edition Edition, Pg:183-191
9. Hillcrest Labs Loop Pointer <http://hillcrestlabs.com/products/loop.php>
10. SIK Experiment Guide for Arduino - V3.2 <https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32/experiment-9-using-a-flex-sensor>
11. Digi: <http://www.digi.com/support/forum/39307/transmit-receive-data-from-two-xbee-zigbee-radios-router-mode>
12. [Agus Kurniawan](#), XBee IEEE 802.15.4 Programming Kindle Edition, Pg: 67-76