**Chapter No: 1**

**INTRODUCTION**

* 1. **Problem Statement:**

In today’s advanced technological world all the equipment are slowly turning towards automation however the craftsmanship of humans will never be matched by that of machines this project is focused on bringing these abilities of humans in to machines. These types of projects will bridge the gap between the movements of machines as close as possible to the movements of humans.

As we all know living beings cannot work in adverse environment like extreme heat, cold, pressure etc. whereas machines are nonliving they can work in any type of environment without the fear of any loss of life.

The problem in controlling the robotic arm were traditional switches which basically work between 2 modes that is on & off, whereas the sensors based controller can work with multiple input and multiple output which increases the degree of control in the system this project has wireless mode for communication between the controller and the system which helps us in controlling while maintaining a distance and viewing in real time.

* 1. **Objective**  This project aims to control the robotic arm assembly using wireless controller based on flex sensor and accelerometer for positioning, picking and placing applications which can be used in various industrial applications for increased safety of human beings working in harsh environment.
  2. **Scope:**

1. The machine will be great use to perform repetitive tasks of picking and placing of small parts in industrial production line with great precision
2. Its use can be extended and exploited by few modifications to difficult and hazardous tasks for industrial applications
3. It can be used to small assembly work effectively due to its great added accuracy for placement of parts which is further extended use of our project.
4. In various industries like the construction it can be use to move big chunky weights from one place to another and place this blocks at the preferred locations.
5. This project can be extended to produce voice information or display the information on LCD based on the hand gesture of the user.

**Chapter No: 2**

**LITERATURE REVIEW:**

Gesture Controlled Mobile Robotic Arm Using Accelerometer in which they have presented a model to control robotic arm through human gestures using accelerometer. A three axis accelerometer is mounted on human hand in order to perform the action of robotic arm according to the action of human hand. Accelerometer is connected to the Atmega 16 Microcontroller which is programmed to take analog readings from accelerometer and transmit them using RF transmitter to the receiving unit at robotic arm. Movements of the robotic arm are achieved through Servo-Motors. The arm is also equipped with a gripper to facilitate the pick and drop facility. The main aim was to control the robotic arm using human gestures wirelessly with smooth movement over a range[1].

“MEMS Accelerometer Based Nonspecific-User Hand Gesture Recognition” in which they have presented three different gesture recognition models which are capable of recognizing seven hand gestures, i.e., *up*, *down*, *left*, *right*, *tick*, *circle*, and *cross*, based on the input signals from MEMS 3-axes accelerometers[2]. The accelerations of a hand in motion in three perpendicular directions are detected by three accelerometers respectively and transmitted to a PC via Bluetooth wireless protocol[2].

“Flex sensor based Nonspecific - user hand gesture recognition” in which they have made a glove with flex sensors. Depending on the movements of finger, the required object is moved. The method ensures effective communication. The system efficiently translates the hand gestures into movement of the objects. The movements of the object were done using DC electric motors [3].

“Implementation of Flex Sensor for Hand gesture based wireless automation of material handling model” in this they have worked on controlling of robot using different sensors like flex sensors, 3 axis accelerometer, ultrasonic sensor and electronic compass. Here flex sensor is mounted on finger and motion is controlled with respect to bending of fingers [4].

“Novel Approaches for Robotic Control Using Flex Sensor” in which they developed the Prosthetic robotic hand using flex sensor for amputees. Their aim was to develop the robotic hand that performs pick and place activities. They have used flex sensors to sense the signals from artificial hand signal are transmitted and that signal is used to drive the mechanical hand. Prosthetic hand must resemble human hand in size and shape and must perform like human hand [5].

**Chapter No: 3**

**PROPOSED METHODOLOGY: WORKING OF MODEL**

This model has two basic components, they are TRANSMITTER and RECEIVER the transmitter is the controller which will be mounted on the hand glove of the operator and the receiver module will be fixed on the near the mechanical assembly

The hand glove has total of three sensors, two flex sensors and one three dimensional accelerometer (ADXL335) the accelerometer will be mounted on the palm of the operator so as soon as the operator moves his hand the accelerometer will sense its position of the hand in 3d space and generate a signal accordingly for the microcontroller.

The flex sensors will be mounted on the index finger and the thumb of the operator. When the operator makes pinching gesture to hold object, the flex sensors resistance will change according to the bending position of the index finger and thumb and it will generate a signal for the microcontroller. This signals from both sensors are transferred through RF transceiver (nRF24L01).

The microcontroller will receive all signals from the sensors and it will process them according to the program. The program will be such that the change of resistance will be proportional to the angle created by servo motor (no.5). This will enable the operator to hold the object of different size and shape with great precisions. In this way the pinching gesture made with by fingers is used for holding action of the mechanical assembly.

The signal generated by the accelerometer will directly tell the microcontroller about the position of the hand of operator in 3d space this signal is processed or ran under program and is used to generate a corresponding signal to rotate servo motor (no1, no2, no3, no4) all this servo motors will rotate simultaneously for the location the targeted objected according to hand movement.

The receiver module has a microcontroller (Nano Arduino) to which all the servo motor are interfaced when the data is received by the receiver module it fed to the microcontroller which generates output signals for all the servo motors present in the mechanical assembly.

All of above process is carried out in real time from the input signals generated from the fingers for pinching gesture and the movement of hand for the location of the target. The input in this whole process is the signals from the hand and finger of the operator and the output can be observed as mechanical arm moves or rotational movement of the servos.

**Chapter No: 4**

**SYSTEM DEVELOPMENT**

**4.1) Block Diagram**

The block diagram of transmitter side is as shown:

Figure 1: Block diagram for transmitter side

The transmitter is used to send the data generated by the sensors i.e. the movement of the hand. For this two sensors are used, MPU and flex sensors which are mounted on the hand glove. The location of the hand is tracked by the MPU sensor, the MPU gives the output with respect to x,y,z axis in the 3-D plane. The MPU is mounted on the hand glove and as we move our hand in space the MPU gives output regarding the x,y,z coordinates [1]. For pinching gesture the flex sensor is mounted on the index finger of hand glove which is used to sense the bending action and thus generate output which is used to create the pinching action at receiver side [5]. So total of four variables are stored in the microcontroller which are x,y,z axis coordinates values and Rv the output of the flex sensor for picking gesture. All these values are transferred from the transmitter with the help of a communication module, in this case used NRF module. The data is sent in the form of packets each packet consisting x,y,z,Rv values the data transfer in the microcontroller is based on FIFO the first in first out mode [3]. In this module as there is no data transfer from receiver to transmitter used only one way communication i.e. from transmitter to receiver. In this way the hand gestures and movements are converted in to signals and transferred.

The block diagram of receiver is shown:

Figure 2: Block diagram for receiver side

At the receiver side the robotic arm assembly consists of servo motors, a microcontroller and receiving module. The receiving module receives the data sent by the transmitter this data sent to the microcontroller which decodes the data and generates the controlling signals for the servo motors accordingly and thus the robotic arm is able to carry out moves like up, down, rotating and pinching action. The block diagram describing the receiver side is shown in Fig 2.

**4.2) Components**

**4.2.1) Microcontroller-LilyPad Arduino**

Microcontroller can be termed as a single on chip computer which includes number of peripherals like RAM, EEPROM, Timers etc., required to perform some predefined task.

The LilyPad Arduino 328 Main Board is an Arduino-programmed microcontroller designed to be easily integrated into e-textiles and wearable projects. The LilyPad Arduino was designed and developed by Leah Buechley and SparkFun Electronics. The LilyPad Arduino consists of an ATmega328 with the Arduino bootloader and a minimum number of external components to keep it as small (and as simple) as possible. This board will run from 2V to 5V and offers large pin-out holes that make it easy to sew and connect. Each of these pins, with the exception of (+) and (-), can control an attached input or output device (like a light, motor, or switch). This LilyPad board has 22 pins — the silver petal-like tabs that ring the outside of the board — twice as many as any other LilyPad Arduino. Each of these pins, with the exception of (+) and (-), can control an attached input or output device (like a light, motor, or switch). We can program this LilyPad board using the Arduino programming environment. The LilyPad Arduino can be powered via the USB connection or with an external power supply. If an external power supply is used, it should provide between 2.7 and 5.5 volts.Dimensions of LilyPad board are:50mm outer diameter**,** thin 0.8mm PCB.

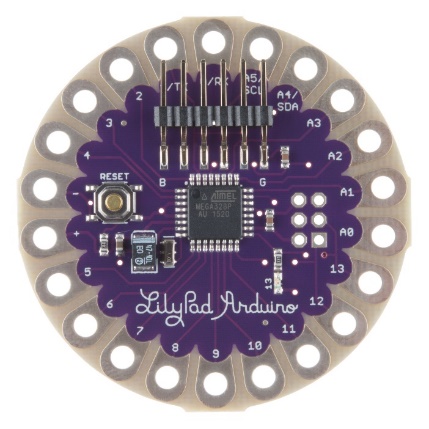


Figure 3: LilyPad Arduino

|  |  |
| --- | --- |
| Microcontroller | ATmega328 |
| Operating Voltage | 2.7-5.5 V |
| Input Voltage | 2.7-5.5 V |
| Digital I/O Pins | 14 |
| PWM Channels | 6 |
| Analog Input Channels | 6 |
| DC Current per I/O Pin | 40 Ma |
| Flash Memory | 32 KB (of which 2 KB used by bootloader) |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Clock Speed | 16 MHz |

Table 1: Specifications of LilyPad

**4.3.2) Microcontroller- Arduino Nano**

Arduino Nano is a surface mount version with integrated USB port. It is a smallest, complete, and breadboard friendly. It comes with 8 analog input pins and onboard +5V AREF. Nano don’t have power jack. The Nano is automatically sense and switch to the higher positive voltage source of power. Arduino Nano got a pin layout that works well with the Basic Stamp (TX, RX, ATN, and GND on one top layer, power and ground on the other). This version comes with Atmel's high performance, low power 8-bit AVR ATMega328 microcontroller in a pint-sized TQFP (Thin Quad Flat Package) package which is brain of this chip. Atmega328P offer more programming and data memory space. The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.

The Nano was designed and is being produced by Gravitech.

The FTDI FT232RL chip on the Nano is only powered if the board is being powered over USB. As a result, when running on external (non-USB) power, the 3.3V output (which is supplied by the FTDI chip) is not available and the RX and TX LEDs will flicker if digital pins 0 or 1 are high.

**Specifications:**

* Microcontroller                        Atmel ATmega328
* Operating Voltage (logic level)  5 V
* Input Voltage (recommended)  7-12 V
* Input Voltage (limits)               6-20 V
* Digital I/O Pins                        14 (of which 6 provide PWM output)
* Analog Input Pins                    8
* DC Current per I/O Pin            40 mA
* Flash Memory                          32 KB (of which 2KB used by bootloader)
* SRAM                                     2 KB
* EEPROM                                 1 KB
* Clock Speed                           16 MHz
* Dimensions                            0.70” x 1.70”

**Features:**

* Automatic reset during program download
* Power OK blue LED
* Green (TX), red (RX) and orange (L) LED
* Auto sensing/switching power input
* Small mini-B USB for programming and serial monitor
* ICSP header for direct program download
* Standard 0.1” spacing DIP (breadboard friendly)
* Manual reset switch

**Description of pins:**

|  |  |  |  |
| --- | --- | --- | --- |
| Pin No | Name | Type | Description |
| 1-2, 5-16 | D0-D13 | I/O | Digital input/output port 0 to 13 |
| 3, 28 | RESET | Input | Reset (active low) |
| 4, 29 | GND | PWR | Supply ground |
| 17 | 3V3 | Output | +3.3V output (from FTDI) |
| 18 | AREF | Input | ADC reference |
| 19-26 | A0-A7 | Input | Analog input channel 0 to 7 |
| 27 | +5V | Output | +5V output (from on-board regulator) |
| or Input | or +5V (input from external power supply) |
| 30 | VIN | PWR | Supply voltage |

Table 2: Input and output pins of Nano

Each of the 14 digital pins on the Nano can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() 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.

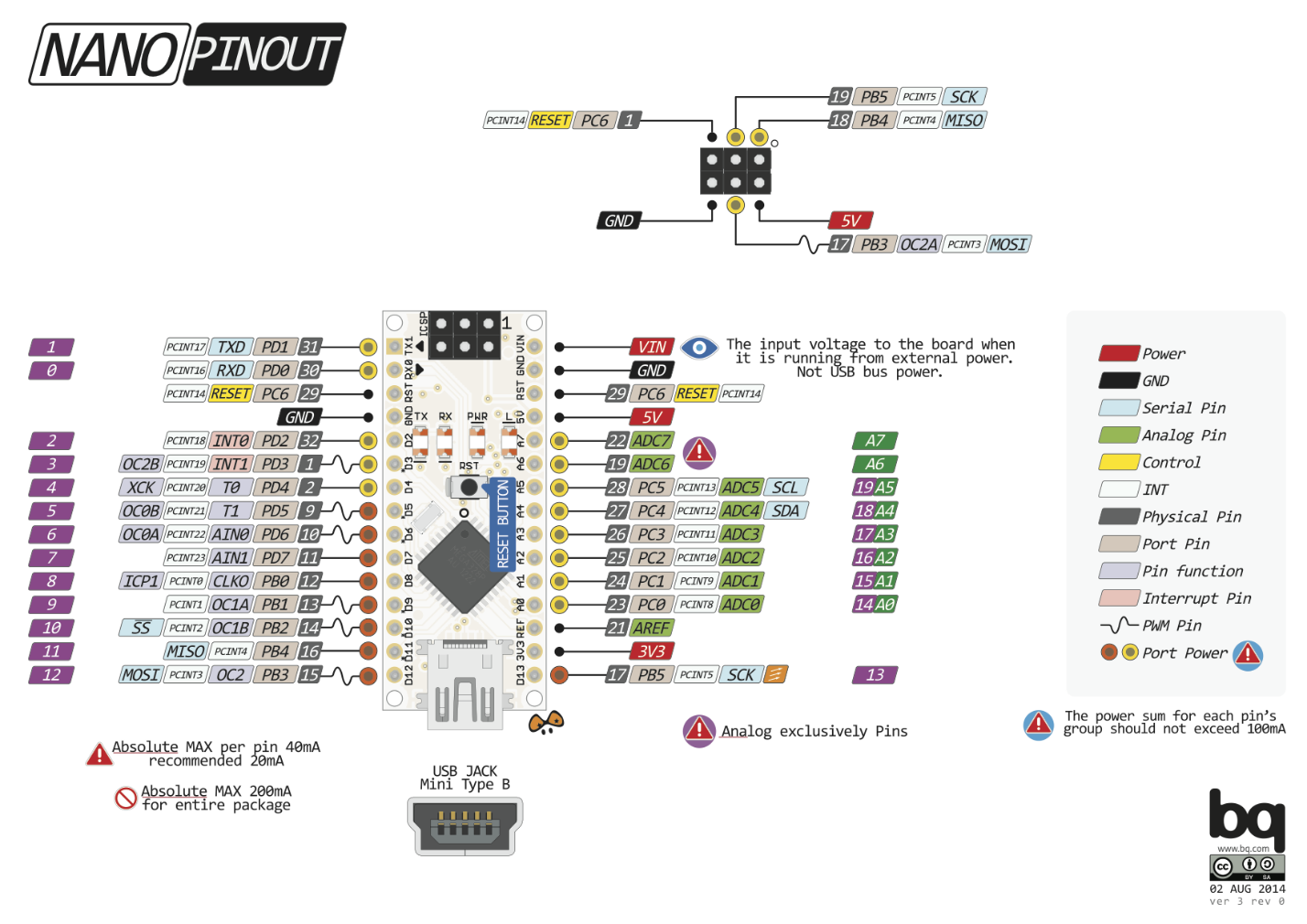


Figure 4: Pin diagram of Arduino Nano

* Serial: 0 (RX) and 1 (TX).

Used to receive (RX) and transmit (TX) TTL(*Transistor–transistor logic*) serial data. These pins are connected to the corresponding pins of the FTDI 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 attach Interrupt () 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, which, although provided by the underlying hardware, is not currently included in the Arduino language.

* 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.

* I2C: 4 (SDA) and 5 (SCL).

Support I2 C (TWI) communication using the Wire library (documentation on the Wiring website). There are a couple of other pins on the board:

* AREF.

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.

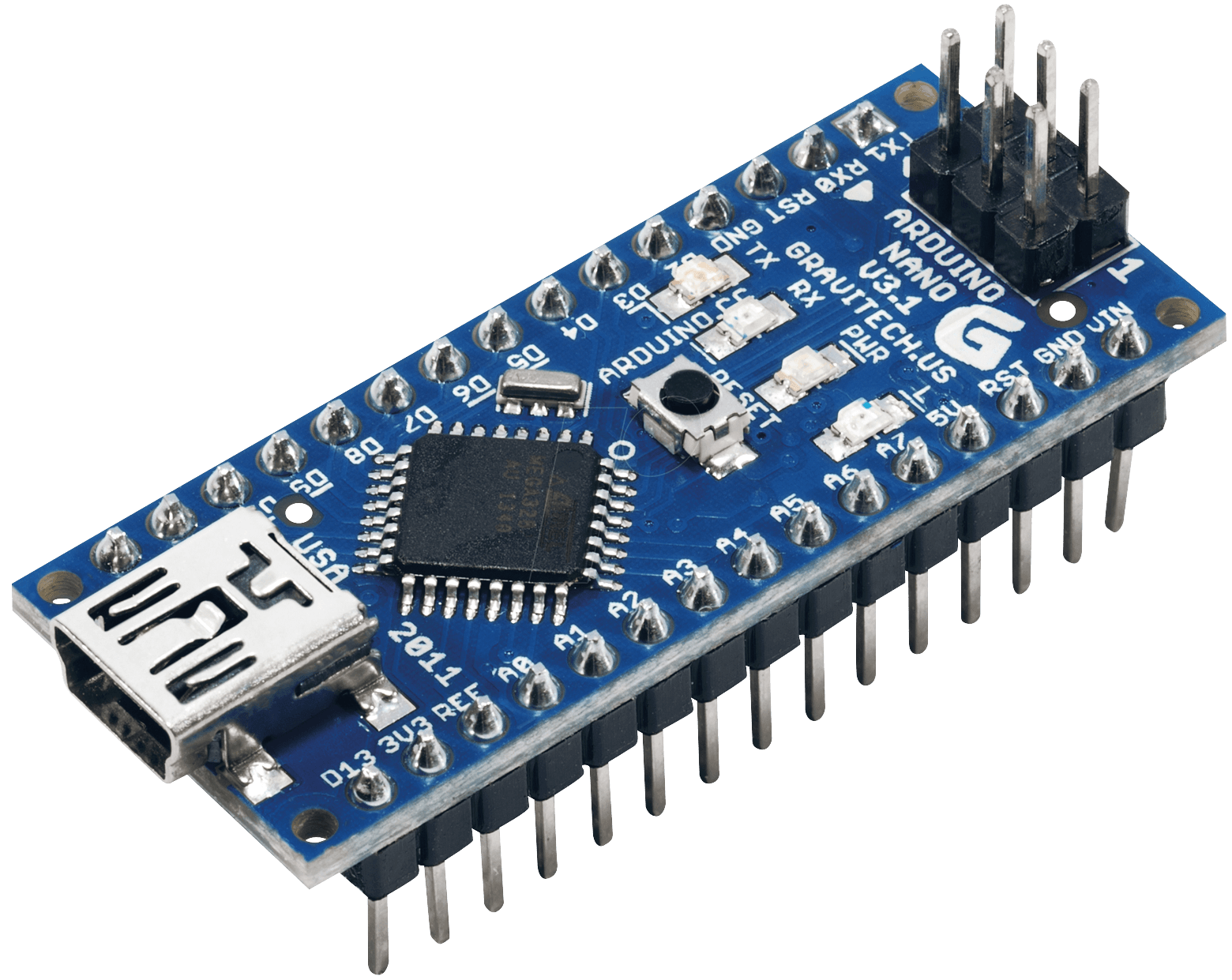


Figure 5: Arduino Nano

**4.2.3) Flex sensor**

This flex sensor is a variable resistor like no other. The resistance of the flex sensor increases as the body of the component bends.A simple flex sensor 2.2" in length. As the sensor is flexed, the resistance across the sensor increases. The resistance of the flex sensor changes when the metal pads are on the outside of the bend (text on inside of bend). Anytime we need to detect a flex, or bend, a flex sensor is probably the part for you. They come in a few different sizes (small, large).

The flex sensor is basically a variable resistor that reacts to bends. Unbent it measures about 22KΩ, to 40KΩ when bend 180º. Note that the bend is only detected in one direction and the reading can be a bit shaky, so we will have best results detecting changes of at least 10º.

**Some applications for the Flex Sensor are:**

* industrial applications, e.g. safety switches, shipping, machine control
* medical applications (e.g. “Smart Bed”)
* gaming devices
* measuring devices
* robotics (e.g. floor mapping, collision detection)
* Physics applications and experiments
* automotive applications

### Properties

* hysteresis/noise negligible
* resistance is function of radius of curvature, not angle at one point
* high temperature and humidity-tolerance
* relatively low cost
* customizable (coatings, laminating materials)

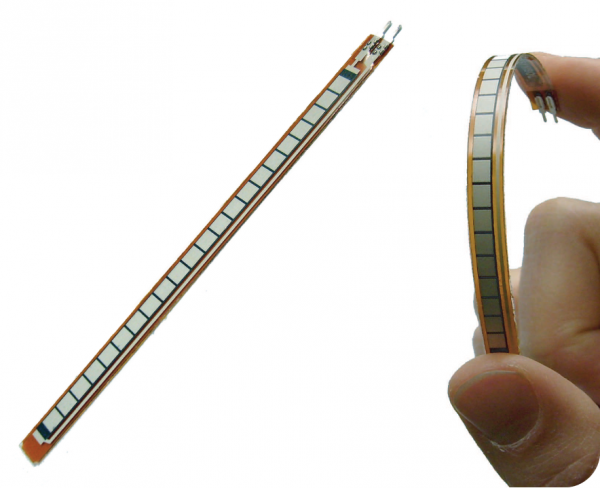
** [](https://cdn.sparkfun.com/assets/learn_tutorials/5/1/1/flex-sensor-direction.png)**

Figure 6: Flex Sensor

**Working**

One side of the sensor is printed with a polymer ink that has conductive particles embedded in it. When the sensor is straight, the particles give the ink a resistance of about 30k Ohms. When the sensor is bent away from the ink, the conductive particles move further apart, increasing this resistance (to about 50k-70K Ohms when the sensor is bent to 90°, as shown in diagram). When the sensor straightens out again, the resistance returns to the original value. By measuring the resistance, you can determine how much the sensor is being bent.

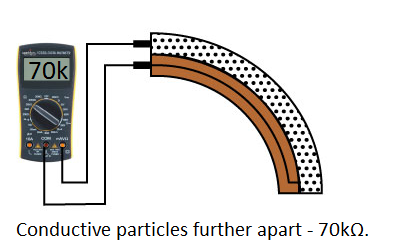
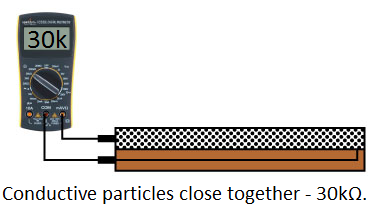
[](https://cdn.sparkfun.com/assets/learn_tutorials/5/1/1/how-it-works-bent.png) [](https://cdn.sparkfun.com/assets/learn_tutorials/5/1/1/how-it-works-straight.png)

Figure 7: Working of Flex Sensor

**4.2.4) Accelerometer- MPU**

MPU-60X0 Overview Motion Interface is becoming a “must-have” function being adopted by Smartphone and tablet manufacturers due to the enormous value it adds to the end user experience. In smart phones, it finds use in applications such as gesture commands for applications and phone control, enhanced gaming, augmented reality, panoramic photo capture and viewing, and pedestrian and vehicle navigation. With its ability to precisely and accurately track user motions, Motion Tracking technology can convert handsets and tablets into powerful 3D intelligent devices that can be used in applications ranging from health and fitness monitoring to location-based services. Key requirements for Motion Interface enabled devices are small package size, low power consumption, high accuracy and repeatability, high shock tolerance, and application specific performance programmability – all at a low consumer price point

The MPU-60X0 is the world’s first integrated 6-axis Motion Tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mm package. The MPU-60X0 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps) and a user-programmable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g.

**Gyroscope Features**

The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features:

• Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full scale range of ±250, ±500, ±1000, and ±2000°/sec

• External sync signal connected to the FSYNC pin supports image, video and GPS synchronization

• Integrated 16-bit ADCs enable simultaneous sampling of gyros

• Enhanced bias and sensitivity temperature stability reduces the need for user calibration

• Improved low-frequency noise performance

• Digitally-programmable low-pass filter

• Gyroscope operating current: 3.6mA

• Standby current: 5µA

• Factory calibrated sensitivity scale factor

• User self-test

**Accelerometer Features**

The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

• Digital-output triple-axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g

• Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer

• Accelerometer normal operating current: 500µA

• Low power accelerometer mode current: 10µA at 1.25Hz, 20µA at 5Hz, 60µA at 20Hz, 110µA at 40Hz

• Orientation detection and signaling

• Tap detection

• User-programmable interrupts

• High-G interrupt

• User self-test

**Three-Axis MEMS Accelerometer**

The MPU-60X0’s 3-Axis accelerometer uses separate proof masses for each axis. Acceleration along a particular axis induces displacement on the corresponding proof mass, and capacitive sensors detect the displacement differentially. The MPU-60X0’s architecture reduces the accelerometers’ susceptibility to fabrication variations as well as to thermal drift. When the device is placed on a flat surface, it will measure 0g on the X- and Y-axes and +1g on the Z-axis. The accelerometers’ scale factor is calibrated at the factory and is nominally independent of supply voltage. Each sensor has a dedicated sigma-delta ADC for providing digital outputs. The full scale range of the digital output can be adjusted to ±2g, ±4g, ±8g, or ±16g.

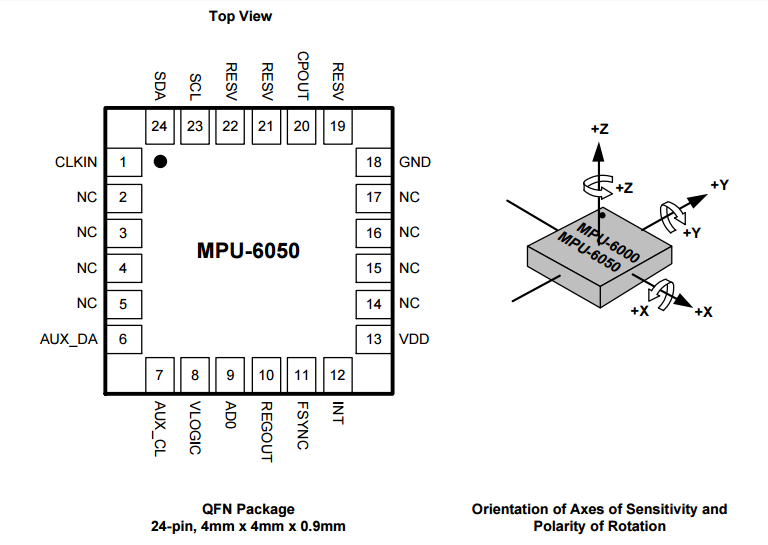
****

Figure 8: MPU Accelerometer

**4.2.5) Communication module**

The module used for communication between transceivers is nRF24LU1+, a single Chip 2.4 GHz Transceiver with USB Microcontroller and Flash Memory. The nRF24LU1+ is a unique single chip solution for compact USB dongles. The internal nRF24L01+ 2.4 GHz RF transceiver supports a wide range of applications including PC peripherals, sports accessories and game peripherals. With an air data rate of 2 Mbps combined with full speed USB, supporting up to 12 Mbps, the nRF24LU1+ meets the stringent performance requirements of applications such as wireless mouse, game controllers and media center remote controls with displays.

The nRF24LU1+ integrates:

* A nRF24L01+ 2.4 GHz RF transceiver
* A full speed USB 2.0 compliant device controller
* An 8-bit microcontroller
* 16 or 32 kbytes of flash memory

All this is packaged on a compact 5x5mm package, low cost external BOM. With an internal voltage regulator that enables the chip to be powered directly from the USB bus, it does not require an external voltage regulator, saving cost and board space. With a fully integrated RF synthesizer and PLL for the USB no external loop filters, resonators or VCO varactor diodes are required. All that is needed is a low cost ±60ppm 16 MHz crystal, matching circuitry and the antenna.

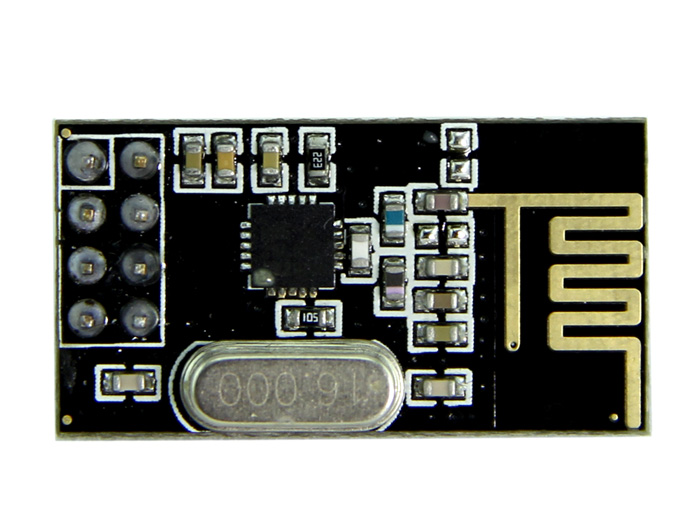


Figure 9: nRF transceiver

The main benefits of nRF24LU1+ are:

* Very compact USB dongle
* Low cost external BOM
* No need for an external voltage regulator
* Single low cost ±60ppm 16 MHz crystal
* Flash memory for firmware upgrades

Key Features

* nRF24L01+ compatible RF transceiver
* Up to 2 Mbps on air data rate
* Up to 12 Mbps USB transfer rate

**4.2.6) Servo motors- MG-996R**

MG996R is an upgraded version of MG995 servo. The new PCB and IC control system which makes it far more accurate. Its internal gearing and motor are also upgraded to improve dead bandwidth and centering. It is a great choice for 1/10 buggy and airplane helicopter, RC-cars from 10 to 6-th scale truggy and monster and many RC model.

Figure 10: Servo Motor

**Specification:**

* Weight: 55g
* Dimension: 40.7×19.7×42.9mm
* Stall torque: 9.4kg/cm (4.8v); 11kg/cm (6.0v)
* Operating speed: 0.19sec/60degree (4.8v); 0.15sec/60degree (6.0v)
* Operating voltage: 4.8~ 6.6v
* Gear Type: Metal gear
* Temperature range: 0- 55deg
* Dead band width: 1us
* servo wire length: 32cm
* servo arms &screws included and fit with Futaba servo arm.

**4.2.7) Battery-LiPo**

These are very slim, extremely light weight batteries based on the new Polymer Lithium Ion chemistry.

A lithium polymer battery, or more correctly lithium-ion polymer battery (abbreviated variously as LiPo, LIP, Li-poly and others), is a rechargeable battery of lithium-ion technology in a pouch format. This is the highest energy density currently in production. Each cell outputs a nominal 3.7V at 2000mAh.

**Technical Details**

* Dimensions: 60mm x 36mm x 7mm / 2.4" x 1.4" x 0.3"
* Weight: 34g
* Nominal Capacity: 2000mAh ±2%
* Nominal Voltage: 3.7V
* Standard Charge Current: ~0.2C / 0.5A
* Charge Cut-Off Voltage: 4.2V
* Standard Discharge Current: ~0.2C / 0.5A

**Features:**

* Excellent long-term self-discharge rates (<5% per month)
* Robust power source under extreme conditions (-25 to 60C)

**Working principle**

LiPos work on the principle of intercalation and de-intercalation of lithium ions from a positive electrode material and a negative electrode material, with the liquid electrolyte providing a conductive medium. To prevent the electrodes from touching each other directly, a micro porous separator is in between which allows only the ions and not the electrode particles to migrate from one side to the other. Do not attempt to charge these with anything but a charger specifically designed for Lithium Ion batteries.



Figure 11: LiPo Battery

**Chapter No: 5**

**HARDWARE IMPLEMENTATION**

**5.1) Controller Side**

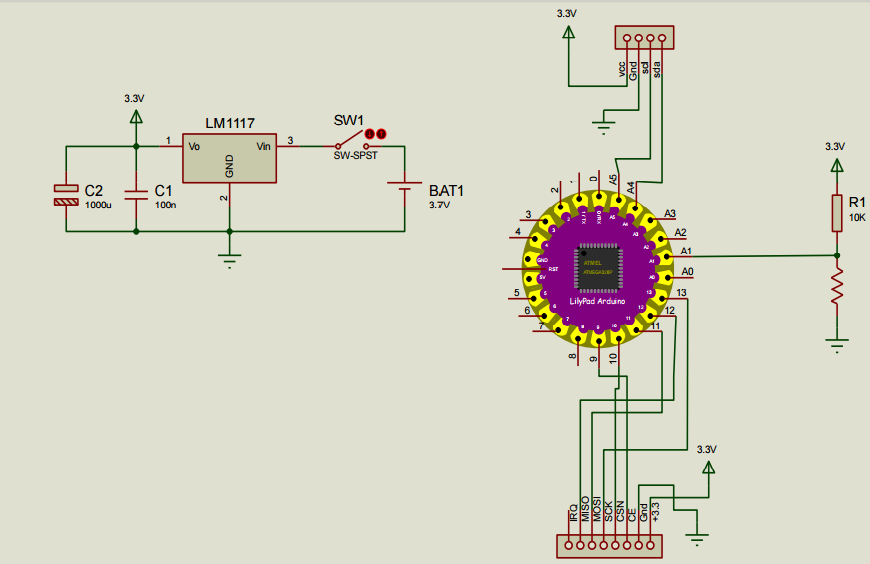
****

Figure 12: Connection diagram for transmitter side

Pin connection between LilyPad and other components

**1) MPU**

* A4- sda
* A5-scl
* Gnd-gnd
* VCC-3.3v

The power to the mpu is feed from the battery output which is 3.3v. The sda and scl pins of the mpu are connected to A4 and A5 pins of the microcontroller. A4 and A5 are analog input pins that is the output from the mpu is analog signal. The ground terminal is connected to common ground. The vvc terminal receives 3.3v supply

**2) Flex sensor**:

Flex sensor is connected to A1 pin of lilypad and to its one end a supply of 3.3v is given and other is grounded via resistor, a output is tapped out from this branch and given to the lilypad.

**3) nRf Communication module**

* Miso-12
* Mosi-11
* Sck-13
* Csn-10
* Ce-9
* Gnd-gnd
* Vcc-3.3v

The MISO pin of the nrf is connected to the 12 no pin the function of MISO pin is serial pheriferal data output bus **MOSI (Master Output/Slave Input)** – Line for the master to send data to the slave. The MOSI pin is connected to 11 no pin the function of this pin is serial pheriferal data input **MISO (Master Input/Slave Output)** – Line for the slave to send data to the master. The sck pin is connected pin 13 no. this pin is serial peripheral clock input which is used to give clock signal during the communication. The CSN pin of nrf is connected to 10 no pin the **CSN (Slave Select/Chip Select)** -Line for the master to select which slave to send data to. The CE pin f nrf is connected to the 9 no pin CE: Chip Enable Activates RX or TX mode. CE = 0 makes the chip to go into Stand-by. The VCC pin is used for the power supply input to the nrf module to this a supply of 3.3v is given.

**4) Power supply:**

For power supply a lithium polymer battery is used. The output from this battery is controlled with lm1117 chip which gives a constant output voltage of 3.7v dc. This power supply circuit also has battery charging functionality. The battery is charged via charging capacitors.

**5.2) Receiver Side**

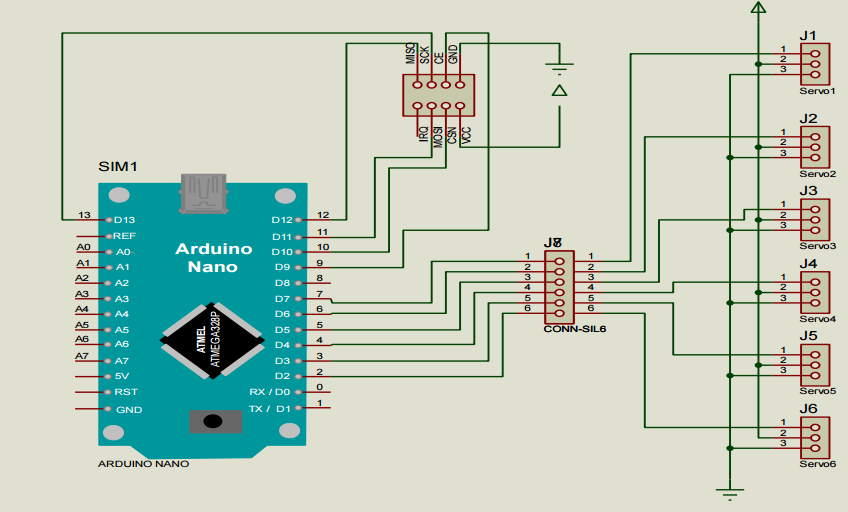


Figure 13: Connection diagram for receiver side

Pin connection between Microcontroller and other components.

**1). nRf Communication Module**

* Miso-12
* Mosi-11
* Sck-13
* Csn-10
* Ce-9
* Gnd-gnd
* Vcc-3.3v

The MISO pin of the nrf is connected to the 12 no pin the function of MISO pin is serial pheriferal data output bus **MOSI (Master Output/Slave Input)** – Line for the master to send data to the slave. The MOSI pin is connected to 11 no pin the function of this pin is serial pheriferal data input **MISO (Master Input/Slave Output)** – Line for the slave to send data to the master. The sck pin is connected pin 13 no this pin is serial pheripheral clock input which is used to give clock signal during the communication. The CSN pin of nrf is connected to 10 no pin the **CSN (Slave Select/Chip Select)** -Line for the master to select which slave to send data to. The CE pin f nrf is connected to the 9 no pin CE : Chip Enable Activates RX or TX mode. CE = 0 makes the chip to go into Stand-by. The VCC pin is used for the power supply input to the nrf module to this a supply of 3.3v is given.

**2) Servo motor interfacing**

Servo motors have three terminals ground supply and control terminal the servo motor motion is carried out depending on the control signal that is the angular displacement is carried out by motor depends on the fashion of control signal pin 7,6,5,3,2 of aurdino generates the required control signal for respective servo motor.

**5.3) Calculation-Equation of Motor Rating**

In this arm there are multiple points. To determine the torque around each joint and to select the appropriate motor, the following described method is used.

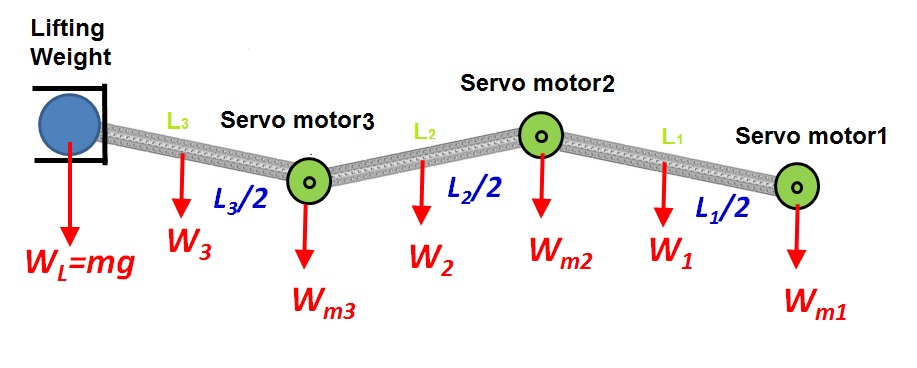


Figure 14: Torque Calculation for motors

Calculation of torque around each motor can be obtained by equation:







Hence for finding the rating of motor, we measured the length of arms and the weight. The arm lengths (in cm.) are L1= 34, L2=30, L3=18, L4=11, L5=7

Let the weight to be lifted be .200 kg. Weight(in grams) of parts of assembly are W1= 380, W2=100, W3=170, W4=290, W5=150

For Motor M1: Torque= Mass X Length.

kg-cm. Therefore the rating of motor 1 must be more than 12.92 kg-cm. We are using 13 Kg-cm.

For Motor M2: Torque= Mass X Length.

kg-cm. Therefore the rating of motor 1 must be more than 12.92 kg-cm. We are using 3.2 Kg-cm.

For Motor M3: Torque= Mass X Length.

kg-cm. Therefore the rating of motor 1 must be more than 12.92 kg-cm. We are using 3.2 Kg-cm.

For Motor M4: Torque= Mass X Length.

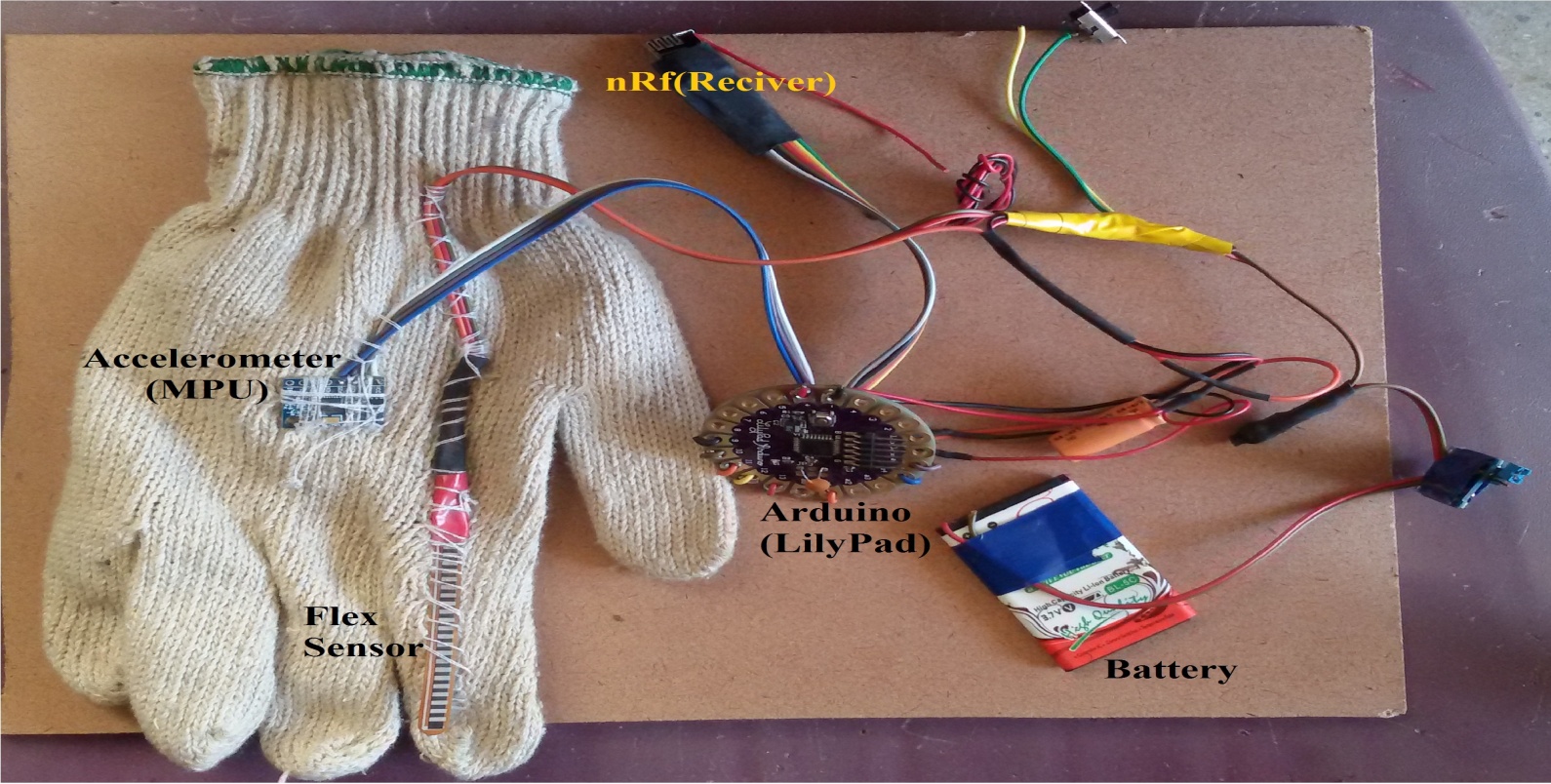
kg-cm. Therefore the rating of motor 1 must be more than 12.92 kg-cm. We are using 3.2 Kg-cm.

For Motor M5: Torque= Mass X Length.

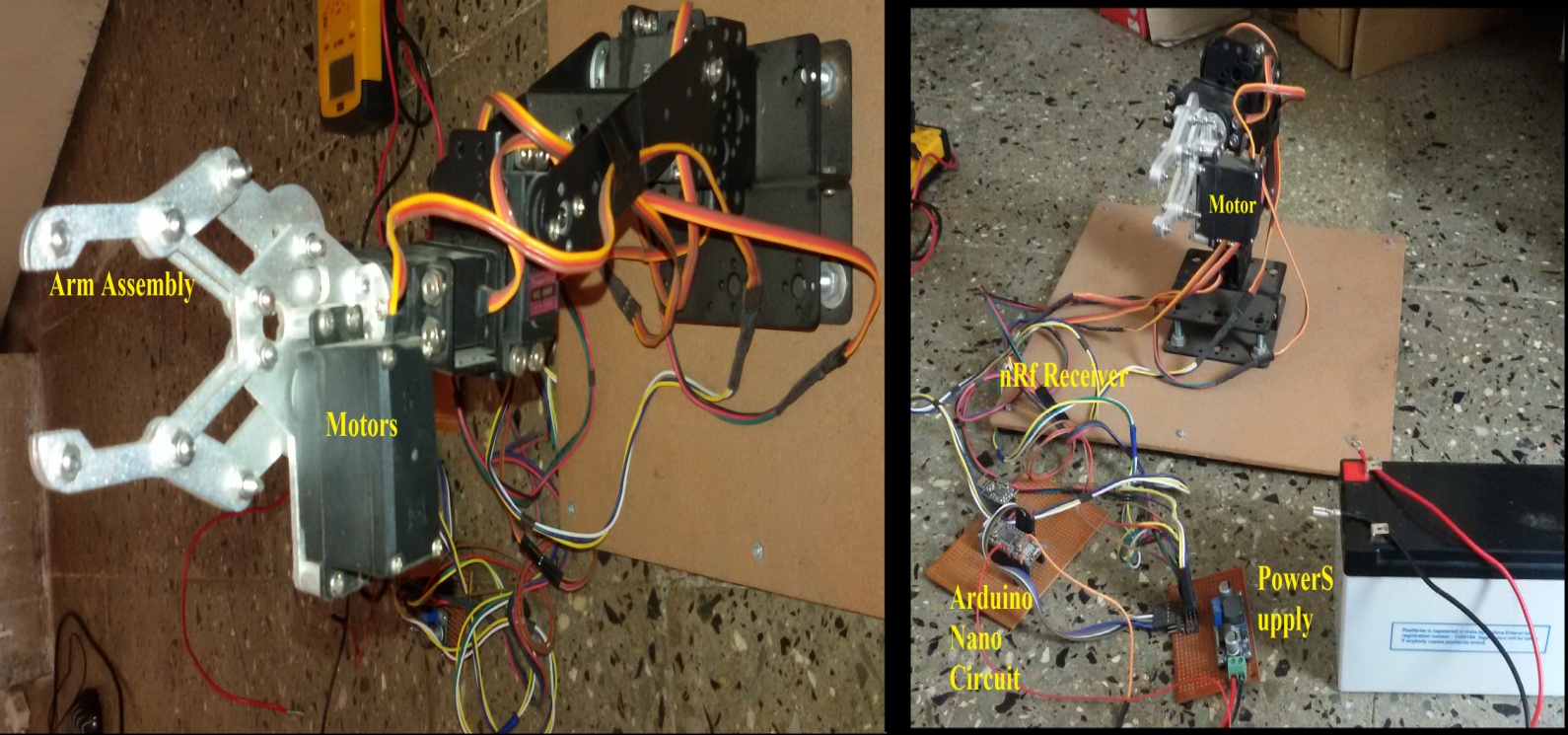
kg-cm. Therefore the rating of motor 1 must be more than 12.92 kg-cm. We are using 3.2 Kg-cm.

**5.4) Photographs of Implementation**

**Controller (Transmitter Side)**

****

**Arm Assembly (Receiver Side)**

****

**Chapter No: 6**

**TESTING & RESULTS**

For flex sensor

The flex sensor output is mapped to motor5

|  |  |  |
| --- | --- | --- |
| Finger bending position | Resistance Ω | Deflection |
| Straight | 210 | 0o |
| Bent at 90o | 300 | 80o |

Table 3: Result Table for flex sensor

For each bending position where resistance is varied by 1.125 Ω there is deflection of 1o. The starting position resistance minimum value is 210 ohm and maximum value is 300ohm and the minimum corresponding deflection is 0o and maximum value is 80o.

For mpu

The mpu6050 is used in three dimension accelerometer mode and its output is maped to the motor1, motor2, motor3, motor4. The mpu is maped in the ratio 1:1 that is for 1cm movement of our hand in any direction the servos will deflect 1o.

|  |  |  |  |
| --- | --- | --- | --- |
| Co-ordinate | Motors | Minimum deflection | Maximum deflection |
| x | Motor1 | 0o | 180o |
| y | Motor2 | 0o | 130o |
| y | Motor3 | 120o | 120o |
| z | Motor4 | 0o | 180o |

Table 4: Result Table for mpu sensor

So for the robotic arm will move exactly the same distance as our arm in respective direction till its maximum value is reached after which it will remain at its max deflected position. As for its claws it will maintain pinching action as long as our finger is bent when we release our finger the robotic arm will also release its claws.

**Chapter No: 7**

**APLLICATIONS**

The wireless control of robotic arm has endless possibility of application it can range from small arms with precise control in the medical application to big size robotic arms used in construction industry.

The robotic arm is now mainly used in the automotive industries to perform various functions such as welding, painting, picking and placing of various parts. The main function of this arm is to reach places of welding where it is difficult for manual welding. The robotic welding is more precise and uniform as compared to welding performed by humans.

In medical applications small robotic arm with precise control could help the surgeons in complex surgeries. Wireless robotic arm could be used to collect blood samples from patients suffering from communicable diseases without any danger of spreading them further.

In construction industry robotic arm and its wireless control has the potential to replace traditional lifting cranes, bull dozers etc. the wireless control will enable the working of these heavy machinery from a safe distance which will increase the work place safety.

The wireless robotic arm could be used by the army to safely diffuse bombs from a safe distance with precise control available which could save many lives. The wireless robotic arm with some predefined operations could help assist handicap people in day to day tasks.

In chemical plants robotic arm could be used to combine harmful chemicals from safe distance to protect the workers from harmful fumes which are emitted during chemical reactions. The robotic arm is more precise in performing chemical reaction as weight or proportion of chemicals is accurately measured.

**Chapter No: 8**

**ADVANTAGES & DISADVANTAGES**

**8.1) Advantages**

1. The advantage of using a motion sensor leads to an easy control of robotic arm
2. The use of complicated mechanisms like pressing of button in analog controller can be avoided
3. Application are performed with precision and high repeatability every time this level of consistency can be hard to achieve any other way
4. With wireless controller for robotic arm through controlling speed increases which directly impacts the applications
5. Robotic arm increases work place safety

**8.2) Disadvantages**

1. The wiring can be very complicated after certain level or at an industrial level
2. The initial investment in the wireless robotic arm is significant
3. The programming of the robotic arm demands proper technical knowledge

**Chapter No: 9**

**CONCLUSION AND FUTURE SCOPE**

The objectives of this project has been achieved which was developing the hardware and Software for an accelerometer controlled robotic arm. From observation that has been made, it clearly shows that its movement is precise, accurate, and is easy to control and user friendly to use. The robotic arm has been developed successfully as the movement of the robot can be controlled precisely. This robotic arm control method is expected to overcome the problem such as placing or picking object that away from the user, pick and place hazardous object in a very fast and easy manner.

There is a lot of room for future scope for this model as of now we have controlled it with simple hand gestures where as in future we could use some predefined voice commands to carry out certain type of movements. Cameras could be added on to the arm assembly for increased visibility when controlling from a distance. Additional sensors could be added to at the arm assembly to sense the target, for some application where color of the target is to be checked such sensors could be added.

For faster processing fast processing microcontrollers could be used as compared to ATmega328. In future the controlling of the arm assembly could be possible not by gestures but by tapping out signals from the muscles or neuro signals which will again make the controlling easier, faster and precise. In future we could control the arm assembly from remote places by transmitting the signals over internet.

**REFERENCES:**

1. Vivek Bhojak, Girish Kumar Solanki, Sonu Daultani “Gesture Controlled Mobile Robotic Arm Using Accelerometer” in International Journal of Innovative Research in Science, Engineering and Technology, Vol. 4, Issue 6, June 2015.
2. Ruize Xu, Shengli Zhou, and Wen J. Li “MEMS Accelerometer Based Nonspecific-User Hand Gesture Recognition” in IEEE SENSORS JOURNAL, VOL. 12, NO. 5, MAY 2012.
3. K.C. Sriharipriya, K.Aarthy, T.S. Keerthana, S.Menaga, S.Monisha “Flex sensor based Nonspecific - user hand gesture recognition” in International Journal of Innovative Research and studies, (ISSN 2319-9725) , Vol. 2, May 2013
4. Dr. Shantanu K. Dixit, Mr. Nitin S. Shingi “Implementation of Flex Sensor for Hand gesture based wireless automation of material handling model” (ISSN 2250-3153), in International Journal of Scientific and Research Publications, Volume 2, Issue 12, December 2012
5. Sangeetha.P, Deepika R, Chethana Gosal S “Novel Approaches for Robotic Control Using Flex Sensor”( ISSN : 2248-9622), in International Journal of Engineering Research and Applications Vol. 5, Issue 2, ( Part -2) February 2015, pp.79-81 [www.ijera.com](http://www.ijera.com)
6. [www.allaboutcircuits.com](http://www.allaboutcircuits.com)
7. www.electrical-engineering-portal.com

**APPENDIX I**

We have used Arduino 1.8 integrated development environments for programming of the microcontrollers on both the receiver side as well as transmitter side. The transmitter side board has direct connection output for program loading whereas for lily pad it was connected via programming unit and the then the program was loaded.

From transmitter side the output of accelerometer (mpu 6050) and the output of the flex sensor. The output of the accelerometer contains X-Y-Z coordinates. The output of the flex sensor is variable resistance. All this data is transferred to receiver side where it is received and processed by the microcontroller and corresponding signal is generated for the servo motors.

**Program for receiver side:**

*#include <SPI.h>*

*#include <nRF24L01.h>*

*#include <RF24.h>*

*RF24 radio(9,10);*

*#include <Servo.h>*

*Servo servo1,servo2,servo3,servo4,servo5; // create servo object to control a servo*

*const byte rxAddr[6] = "00001";*

*int pos1=90;*

*int pos2=30;*

*int pos3=180;*

*int pos4=90;*

*int pos5=30;*

*void setup()*

*{*

*while (!Serial);*

*Serial.begin(9600);*

*radio.begin();*

*radio.openReadingPipe(0, rxAddr);*

*radio.startListening();*

*servo1.attach(2);*

*servo2.attach(3);*

*servo3.attach(4);*

*servo4.attach(5);*

*servo5.attach(6);*

*servo1.write(pos1);*

*servo2.write(pos2);*

*servo3.write(pos3);*

*servo4.write(pos4);*

*servo5.write(pos5);*

*}*

*void loop()*

*{*

*if (radio.available())*

*{*

*char text[32] = {0};*

*radio.read(&text, sizeof(text));*

*String Data(text);*

*Serial.print("x: ");*

*String x=getStringPartByNr(Data, ',', 0);*

*String y=getStringPartByNr(Data, ',', 1);*

*String z=getStringPartByNr(Data, ',', 2);*

*String a=getStringPartByNr(Data, ',', 3);*

*x.remove(0, 1);*

*y.remove(0, 1);*

*z.remove(0, 1);*

*a.remove(0, 1);*

*Serial.print("x: ");*

*Serial.print(x); // the first part - return "192"*

*Serial.print("\t");*

*Serial.print("y: ");*

*Serial.print(y); // the 2. part - return "168"*

*Serial.print("\t");*

*Serial.print("z: ");*

*Serial.print(z); // the 3. part - return "10"*

*Serial.print("\t");*

*Serial.print("a: ");*

*Serial.print(a); // the 4. part - return "145"*

*Serial.print("\t");*

*int cx=x.toInt();*

*int cy=y.toInt();*

*int cz=z.toInt();*

*int aa=a.toInt();*

*if(aa>300)*

*aa=300;*

*if(cx<0)*

*{*

*int ctx=-cx;*

*if(ctx>90)*

*cx=180;*

*else*

*cx=0;*

*}*

*cx = map(cx, 0, 180, 180, 0);*

*if(cy<0)*

*{*

*cy=0;*

*}*

*cy = map(cy, -30, 70, 130, 0);*

*Serial.print("cx: ");*

*Serial.print(cx); // the first part - return "192"*

*Serial.print("\t");*

*Serial.print("cy: ");*

*Serial.print(cy); // the 2. part - return "168"*

*Serial.print("\t");*

*Serial.print("cz: ");*

*Serial.print(cz); // the 3. part - return "10"*

*Serial.print("\t");*

*// else*

*// cy=70-cy;*

*//cy=cy+20;*

*// cx=180-cx;*

*int temp=290-aa;*

*if(temp>80)*

*temp=80;*

*else if(temp<0)*

*temp=0;*

*temp=temp+30;*

*Serial.print("angle: ");*

*Serial.println(temp); // the 4. part - return "145"*

*//if(cz<0)*

*// cz=cz\*-1;*

*cz=cz+100;*

*servo4.write(cz);*

*servo5.write(temp);*

*servo1.write(cx);*

*servo2.write(cy);*

*servo3.write(120);*

*}*

*}*

*String getStringPartByNr(String data, char separator, int index)*

*{*

*// spliting a string and return the part nr index*

*// split by separator*

*int stringData = 0; //variable to count data part nr*

*String dataPart = ""; //variable to hole the return text*

*for(int i = 0; i<data.length()-1; i++) { //Walk through the text one letter at a time*

*if(data[i]==separator) {*

*//Count the number of times separator character appears in the text*

*stringData++;*

*}else if(stringData==index) {*

*//get the text when separator is the rignt one*

*dataPart.concat(data[i]);*

*}else if(stringData>index) {*

*//return text and stop if the next separator appears - to save CPU-time*

*return dataPart;*

*break;*

*}*

*}*

*//return text if this is the last part*

*return dataPart;*

**Program for transmitter side:**

*#include "I2Cdev.h"*

*#include "MPU6050\_6Axis\_MotionApps20.h"*

*MPU6050 mpu;*

*// MPU control/status vars*

*bool dmpReady = false; // set true if DMP init was successful*

*uint8\_t mpuIntStatus; // holds actual interrupt status byte from MPU*

*uint8\_t devStatus; // return status after each device operation (0 = success, !0 = error)*

*uint16\_t packetSize; // expected DMP packet size (default is 42 bytes)*

*uint16\_t fifoCount; // count of all bytes currently in FIFO*

*uint8\_t fifoBuffer[64]; // FIFO storage buffer*

*// ================================================================*

*#include <SPI.h>*

*#include <nRF24L01.h>*

*#include <RF24.h>*

*RF24 radio(9,10);*

*const byte rxAddr[6] = "00001";*

*// ================================================================*

*// ================================================================*

*// === INTERRUPT DETECTION ROUTINE ===*

*// ================================================================*

*volatile bool mpuInterrupt = false; // indicates whether MPU interrupt pin has gone high*

*void dmpDataReady() {*

*mpuInterrupt = true;*

*}*

*// ================================================================*

*// === INITIAL SETUP ===*

*// ================================================================*

*void setup() {*

*// ================================================================*

*radio.begin();*

*radio.setRetries(15, 15);*

*radio.openWritingPipe(rxAddr);*

*radio.stopListening();*

*// ================================================================*

*// join I2C bus (I2Cdev library doesn't do this automatically)*

*#if I2CDEV\_IMPLEMENTATION == I2CDEV\_ARDUINO\_WIRE*

*Wire.begin();*

*TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)*

*#elif I2CDEV\_IMPLEMENTATION == I2CDEV\_BUILTIN\_FASTWIRE*

*Fastwire::setup(400, true);*

*#endif*

*Serial.begin(115200);*

*mpu.initialize();*

*devStatus = mpu.dmpInitialize();*

*mpu.setXGyroOffset(220);*

*mpu.setYGyroOffset(76);*

*mpu.setZGyroOffset(-85);*

*mpu.setZAccelOffset(1788); // 1688 factory default for my test chip*

*// make sure it worked (returns 0 if so)*

*if (devStatus == 0) {*

*mpu.setDMPEnabled(true);*

*attachInterrupt(0, dmpDataReady, RISING);*

*mpuIntStatus = mpu.getIntStatus();*

*dmpReady = true;*

*packetSize = mpu.dmpGetFIFOPacketSize();*

*} else {*

*}*

*}*

*int lx,ly,lz=0;*

*int cx,cy,cz=0;*

*// ================================================================*

*// === MAIN PROGRAM LOOP ===*

*// ================================================================*

*void loop() {*

*// if programming failed, don't try to do anything*

*if (!dmpReady) return;*

*// reset interrupt flag and get INT\_STATUS byte*

*mpuInterrupt = false;*

*mpuIntStatus = mpu.getIntStatus();*

*// check for overflow (this should never happen unless our code is too inefficient)*

*if ((mpuIntStatus & 0x10) || fifoCount == 1024) {*

*// reset so we can continue cleanly*

*mpu.resetFIFO();*

*Serial.println(F("FIFO overflow!"));*

*// otherwise, check for DMP data ready interrupt (this should happen frequently)*

*} else if (mpuIntStatus & 0x02) {*

*while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();*

*// read a packet from FIFO*

*mpu.getFIFOBytes(fifoBuffer, packetSize);*

*fifoCount -= packetSize;*

*#ifdef OUTPUT\_READABLE\_YAWPITCHROLL*

*// display Euler angles in degrees*

*mpu.dmpGetQuaternion(&q, fifoBuffer);*

*mpu.dmpGetGravity(&gravity, &q);*

*mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);*

*cx=ypr[0] \* 180/M\_PI;*

*cy=ypr[1] \* 180/M\_PI;*

*cz=ypr[2] \* 180/M\_PI;*

*int a = analogRead(A1);*

*Serial.print("current\t");*

*Serial.print(cx);*

*Serial.print("\t");*

*Serial.print(cy);*

*Serial.print("\t");*

*Serial.print(cz);*

*String data ="x";*

*data= data + String(cx);*

*data= data + ",y";*

*data= data + String(cy);*

*data= data + ",z";*

*data= data + String(cz);*

*data= data + ",a";*

*data= data + String(a);*

*data= data + ",";*

*Serial.println(data);*

*char text[32];*

*data.toCharArray(text,32);*

*Serial.println(text);*

*radio.write(&text, sizeof(text));*

*#endif*

*}*

*}*