ABSTRACT

A wireless data glove was developed to control a Robotic arm. Sensors mounted on the glove send signals to a processing unit, worn on the user's forearm that translates hand postures into data. An RF transceiver, also mounted on the user, transmits the encoded signals representing the hand postures and dynamic gestures to the robot via RF link. Commands to control the robot's position, camera, claw, and arm include "activate mobility," "hold on," and "grab object.".

Table Of Contents

List Of Figures

CHAPTER 1

INTRODUCTION

Ever since electronics was discovered, it has become an integral part of our life. Over the years control of electronic devices such as smartphones and robots have seen groundbreaking research. Nowadays we are moving towards automation where electronic devices can think on their own.

However sometimes we need a human to control the robot instead of relying on a piece of code. This control of robots may be achieved by either wired or wireless method. Both have their pros and cons. Among these, the recent method of gesture control has become quite popular. The reason behind it being that human hand gestures are natural and hence give a more intuitive form of interaction with the robot. By making this interface wireless using radio frequency, it becomes possible to control the robot very easy and user friendly.

Nowadays number of different wireless robots are being manufactured and used in different fields of industry. This paper describes a wireless gesture control for robots using AtMega328 microcontroller, AtMega2560 microcontroller NRF24L01 wireless module, A gesture sensing glove (consisting of a accelerometer, flex sensors and finger contacts) and a two wheel drive robot.

In the present work, the glove uses accelerometer for sensing the motion of the wrist, the flex sensors for motion of fingers and thumb and finger contacts for sensing the gesture of touching two or more fingers. A single or combination of above motions is used as a gesture which is transmitted by the glove using NRF transceiver to the robot unit, which processes that signal to decide how to move based on the gesture made The robot uses 2 PMDC motors driven by L298n dual Hbridge motor driver module and controlled by another AtMega328 microcontroller.

This gesture control technique has been applied for different purposes. For example in medical field it can be used by disabled patients to send messages by doing a subtle gesture, it can also be used to control all the electronic devices such as LEDs, fans, TV, Air conditioners etc, it can help in giving feedback in Augmented reality devices, it can also be paired with a smart phone to provide a more interactive platform to control and interact with it, it can also be used to control a modern electric vehicle via pairing it to the master control of that vehicle using HumanMachine-Interface technology and hidden Markov method and it can be used as an interactive input device for computers.

1.1 AIMS AND OBJECTIVE

Gesture means the movement of hand and face of humans. The main objective of this project is to control the robotic arm using human gestures. The human gestures are sensed with the help of an accelerometer, also known as inertial sensor. A microcontroller is used in the transmitter section. It is coded in such a way that the required actions for the human gesture are done. These sensed signals are processed and then transmitted to the robotic arm at the receiver section using RF transceiver module. Thus the robotic arm performs the required movement. A remote control system is also used to control the movement of the robot. This system is also uses an RF transceiver module for the wireless communication. The model can be constructed and the required work can be done. Thus, this proposed model will be helpful and avoid danger for the people working in hazardous areas.

1.2 BACK GROUND OF THE PROJECT

The software application and the hardware implementation help the micro controller read the output of the sensors.The project proposes a autonomous robotic arm, In which no remote is used for controlling the robotic actions. It intelligently detects of the hand through the sensors, avoid it and take decision on the basis of internal code that we set for it. The detail information is given in the following subtopics which will help you to understand the whole system and its design.

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

CHAPTER 2

PROJECT DESCRIPTION

2.1 FLOW CHART

The flow chart of the design is as shown in Fig 2.1.The brief description of each unit is explained below.

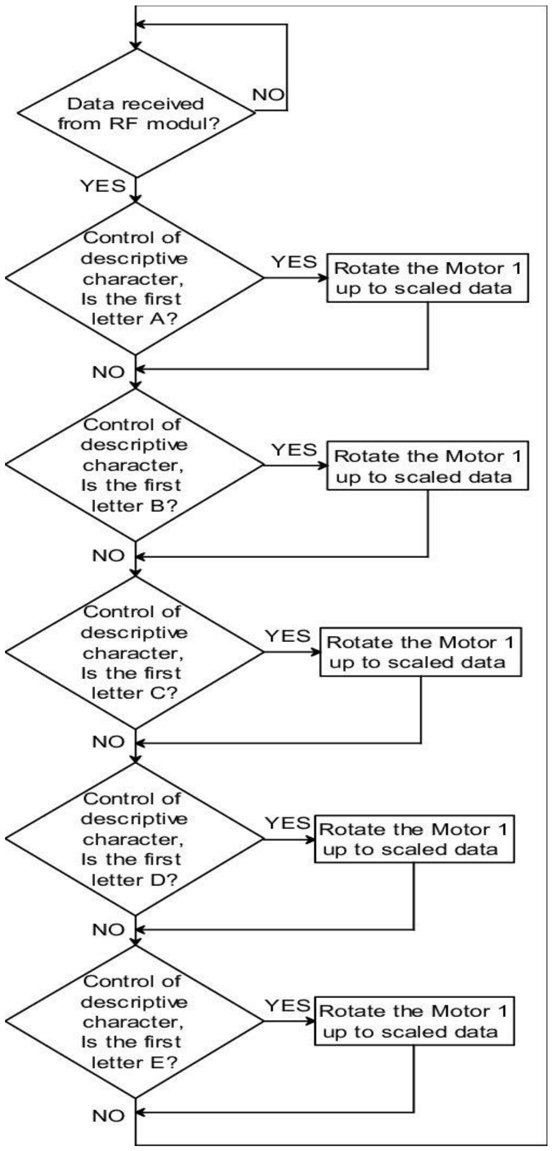

Data received from RF modul?
NO
YES
Control of descriptive character, Is the first letter A?
YES
Rotate the Motor 1 up to scaled data
NO
Control of descriptive character, Is the first letter B?
YES
Rotate the Motor 1 up to scaled data
NO
Control of descriptive character, Is the first letter C?
YES
Rotate the Motor 1 up to scaled data
NO
Control of descriptive character, Is the first letter D?
YES
Rotate the Motor 1 up to scaled data
NO
Control of descriptive character, Is the first letter E?
YES
Rotate the Motor 1 up to scaled data
NO

Fig 2.1: Flowchart for Robotic Arm

2.2 CIRCUIT DIAGRAM

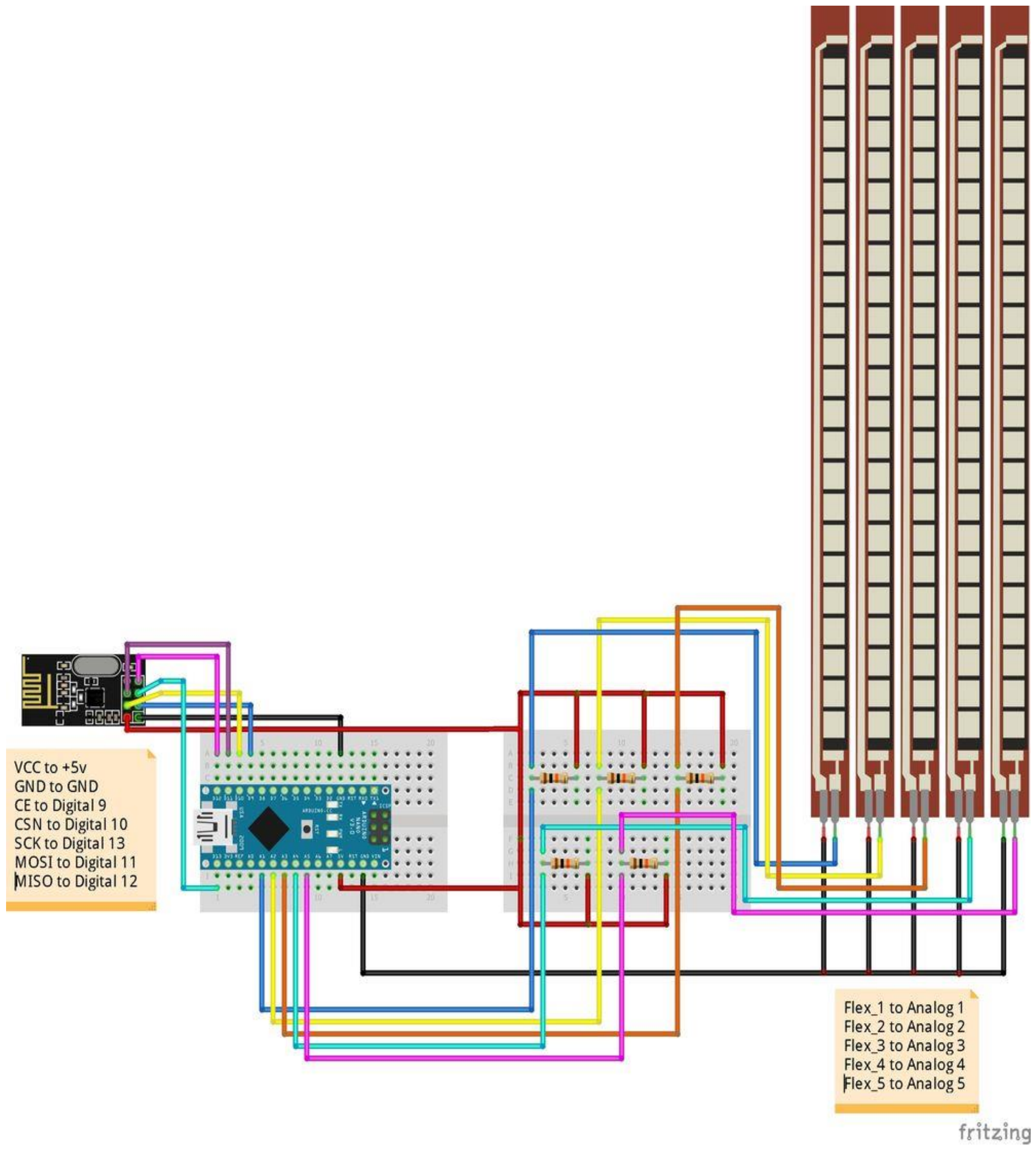


Fig 2.2(a) Circuit Diagram for Transmitter side

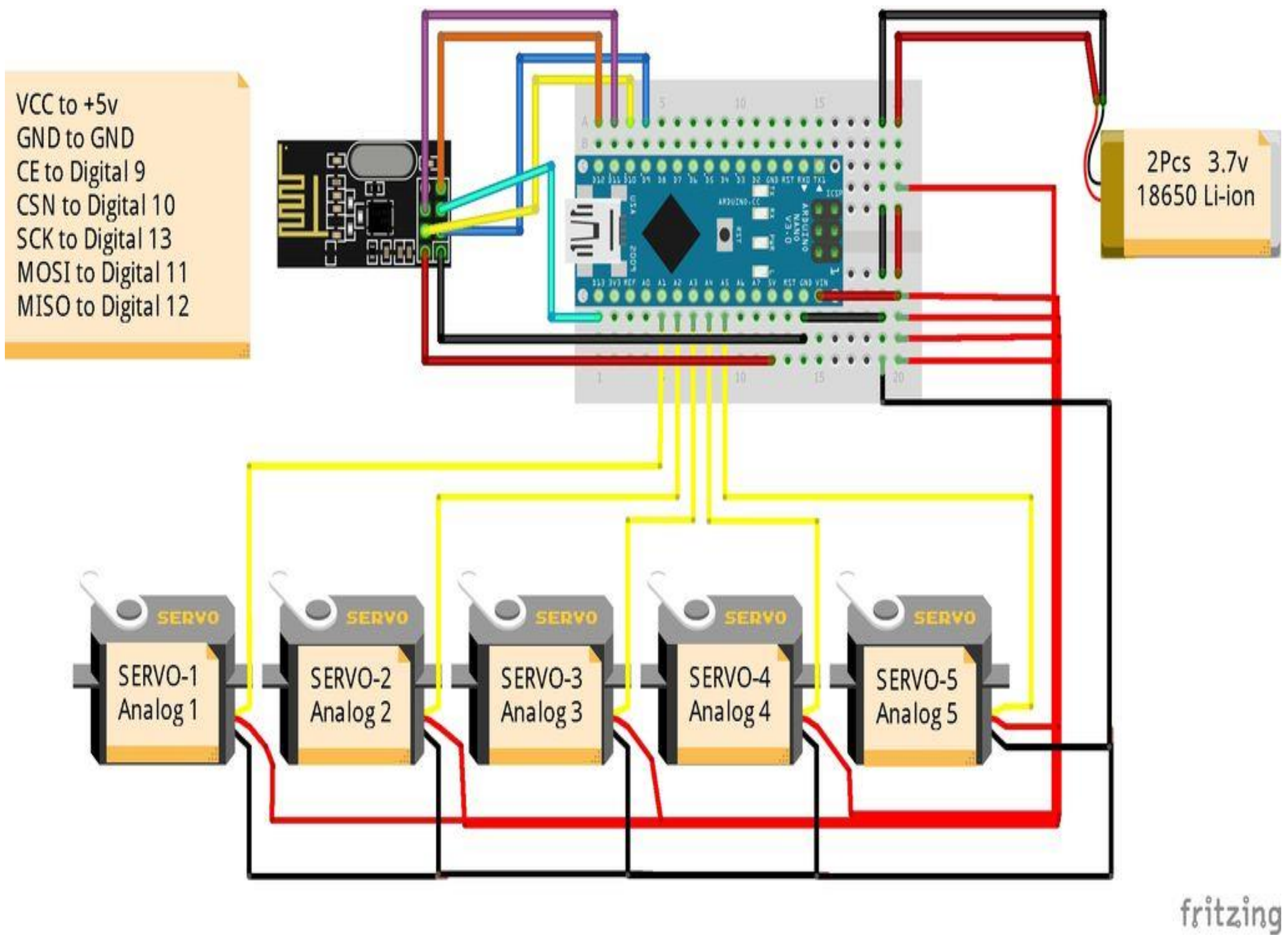


Fig 2.2(b) Circuit Diagram for Receiver side

2.3 WORKING PROCEDURE

Download the RF24.h library and move it to the Arduino libraries folder. https://github.com/maniacbug/RF24

After the flex sensors are connected o the glove, read and note the minimum and maximum values that each flex sensor has detected.

Then enter these values into the transmitter (glove) code.

Keep all servo motors at 10 or 170 degrees before attaching the servo pulleys to the servo motors.

When mounting the servo pulleys, keep fingers in the closed or opened position (according to your servo positions).

Then wrap around the servo pulley until the braid wires becomes stretched.

Move all the fingers to the closed and opened position by checking the servo motors one by one.

Then get the best angles for servo motors (servo angles while fingers closed and opened).

Enter the servo motors angles and flex sensor values to the transmitter code as follows.

flex sensor min. value, flex sensor max. value, servo min. angle, servo max. angle

(flex_val = map (flex_val, 630, 730, 10, 170);

There is only one change in the receiver source code. Which flex sensor in the transmitter will control which servo motor in the receiver? For example, msg[0] sends the data of the x sensor-5. If you want to control the servo motor-5 with the flex sensor-5, you can do this by typing

'servo5.write(msg[0])'.

If you used different pins than pins shown in circuit, change them in both codes.

CHAPTER 3
MICROCONTROLLER

3.1. A brief history of ARDUINO UNO:

Arduino/Genuino Uno is a micro controller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the micro controller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

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

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

VIN.The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin. □

5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it. □

3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA. □ GND. Ground pins.

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

Each of the 14 digital pins on the Uno 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. In addition, some pins have specialized functions: □

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

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

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function. □ SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library. □

LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off. The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function.

Additionally, some pins have specialized functionality: □

TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library. There are a couple of other pins on the board:

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

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other micro controllers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials. The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files). You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by: On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and

Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip.

The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following halfsecond or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened.

If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

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

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

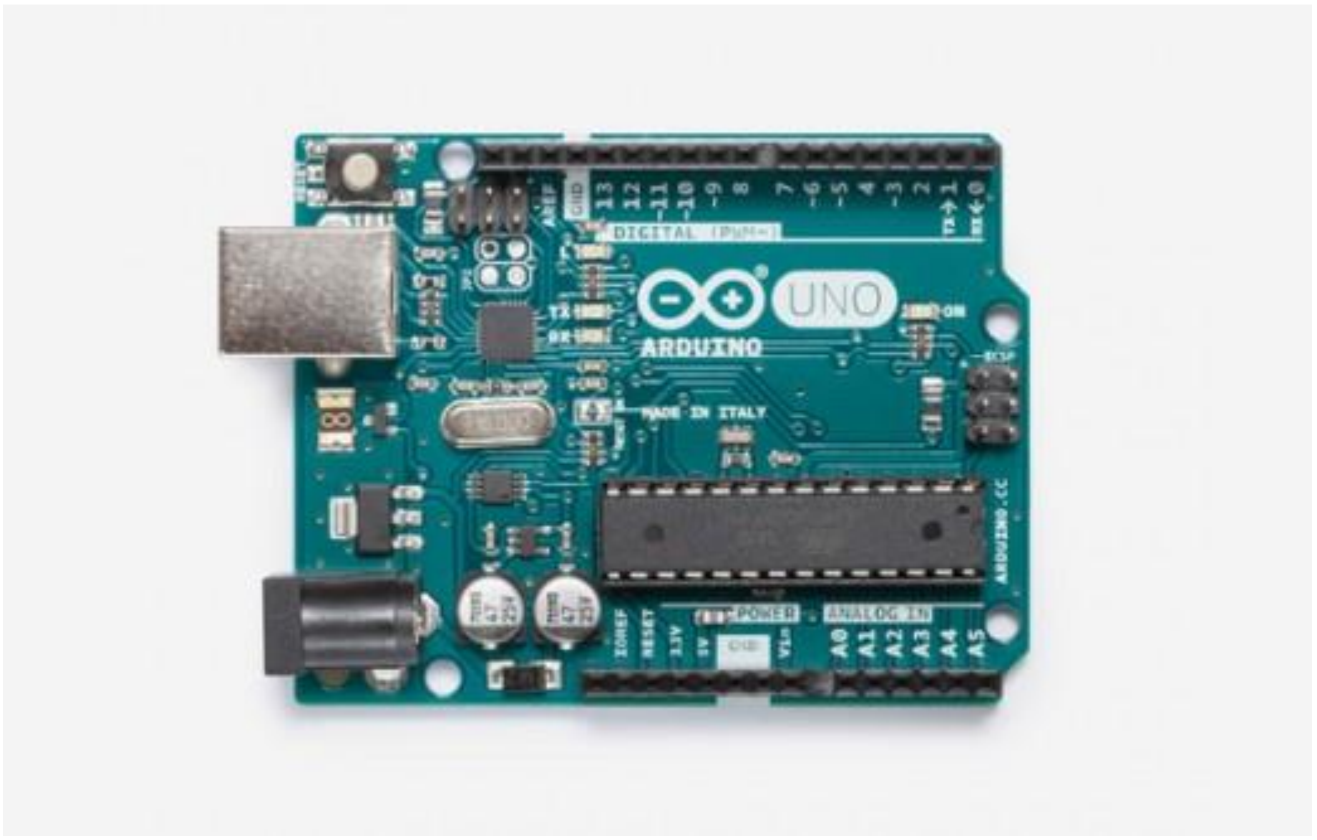


Fig 3.1: Arduino Uno Circuit Board

KEY PARAMETERS

Parameter	Value
Micro Controller	ATmega328
Operating Voltage	5V
Flash memory	32 Kb

SRAM	2 Kb
EEPROM	1 Kb
Pin count	14 Digital I/O pins, 6 Analog Input pins.
Input Voltage(recommended)	7-12 V
Input Voltage(limits)	6-20 V
Clock Speed	16 MHz
External interrupts	-
USB Interface	No

3.2 PIN DIAGRAM:

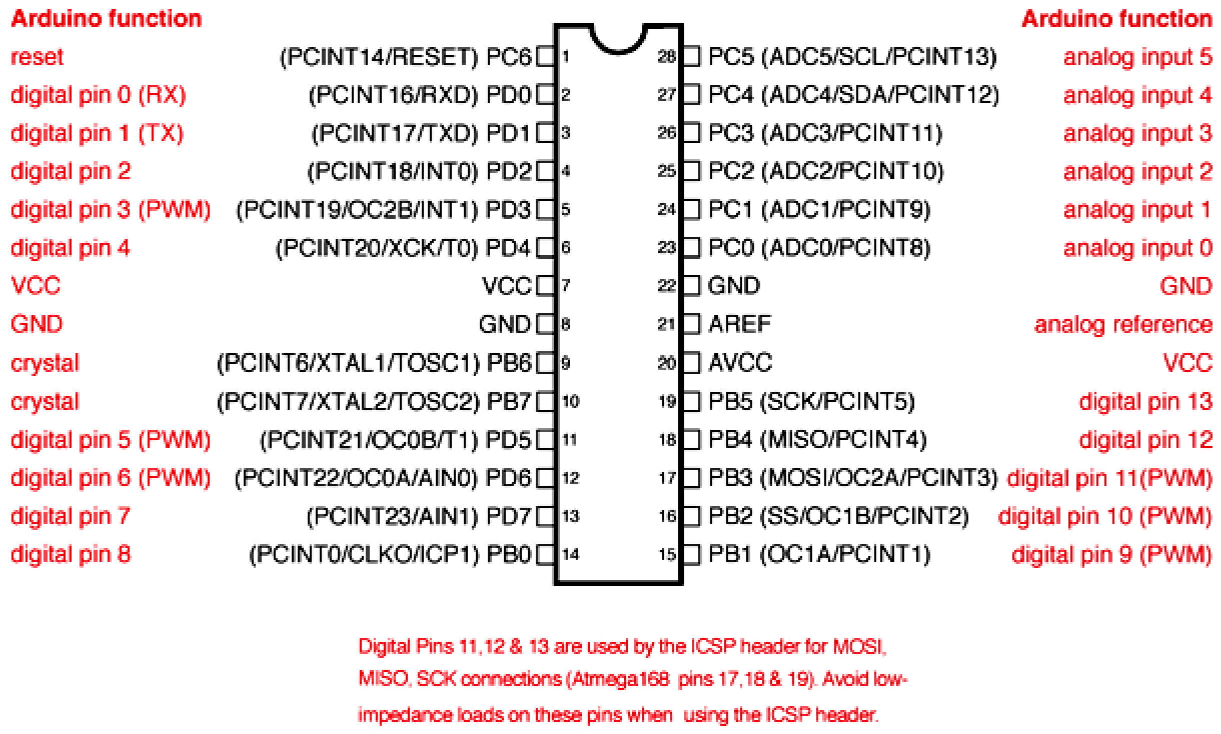


Fig 3.2 Pin Diagram of ARDUINO UNO

3.3 PIN DESCRIPTION

The given below gives a description for each of the pins, along with their function.

Input and Output

Each of the 14 digital pins on the Arduino Uno 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.
In addition, some pins have specialized functions:

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

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

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

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

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

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

There are a couple of other pins on the board:

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.

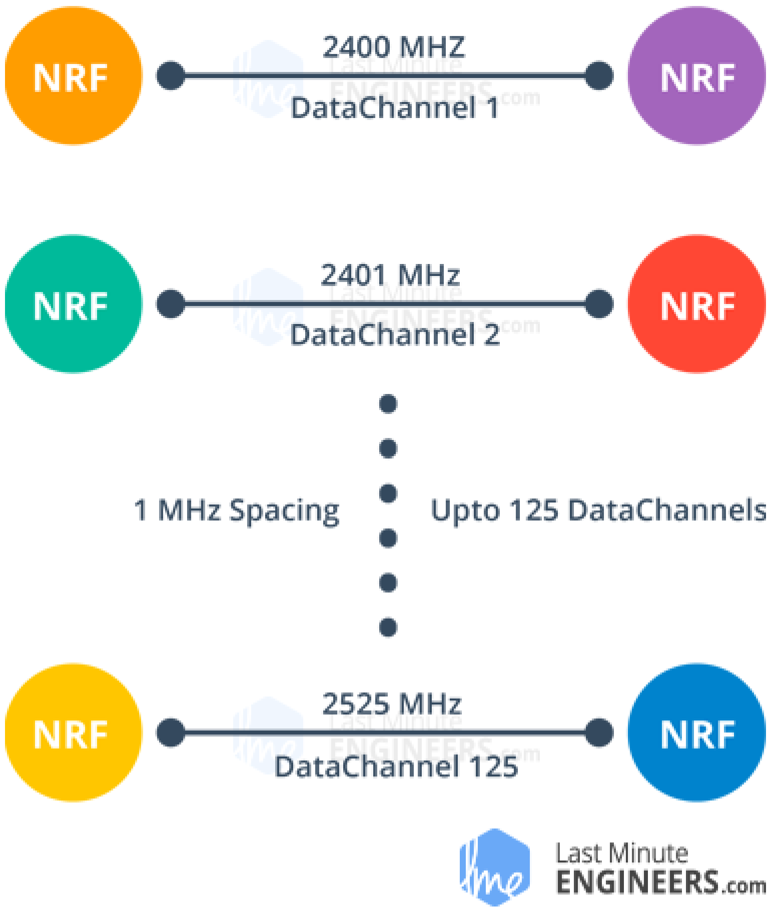
CHAPTER 4

Radio Transreceiver

RF Channel Frequency

The nRF24L01+ transceiver module transmits and receives data on a certain frequency called Channel. Also in order for two or more transceiver modules to communicate with each other, they need to be on the same channel. This channel could be any frequency in the 2.4 GHz ISM band or to be more precise, it could be between 2.400 to 2.525 GHz (2400 to 2525 MHz).

Each channel occupies a bandwidth of less than 1MHz. This gives us 125 possible channels with 1MHz spacing. So, the module can use 125 different channels which give a possibility to have a network of 125 independently working modems in one place.



NOTE

The channel occupies a bandwidth of less than 1MHz at 250kbps and 1Mbps air data rate. However at 2Mbps air data rate, 2MHz bandwidth is occupied (wider than the resolution of RF channel frequency setting). So, to ensure non-overlapping channels and reduce cross-talk in 2Mbps mode, you need to keep 2MHz spacing between two channels.

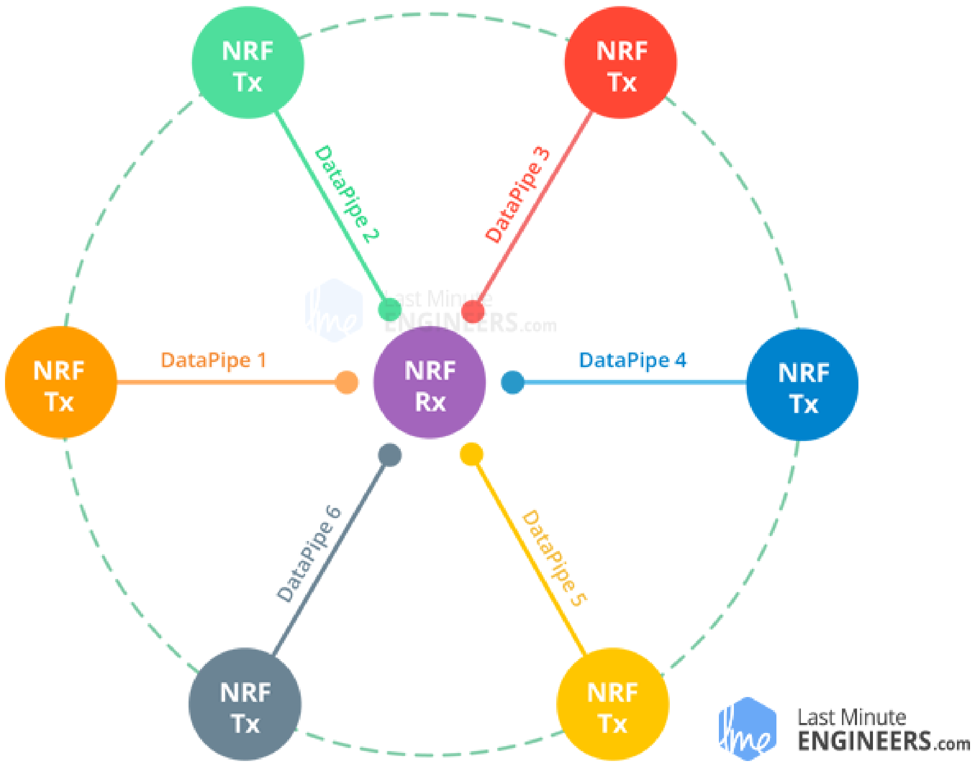
RF channel frequency of your selected channel is set according to the following formula:

$\text{Freq}_{(\text{Selected})} = 2400 + \text{CH}_{(\text{Selected})}$

For example, if you select 108 as your channel for data transmission, the RF channel frequency of your channel would be 2508MHz (2400 + 108)

nRF24L01+ Multiceiver Network

The nRF24L01+ provides a feature called Multiceiver. It's an abbreviation for Multiple Transmitters Single Receiver. In which each RF channel is logically divided into 6 parallel data channels called Data Pipes. In other words, a data pipe is a logical channel in the physical RF Channel. Each data pipe has its own physical address (Data Pipe Address) and can be configured. This can be illustrated as shown below.

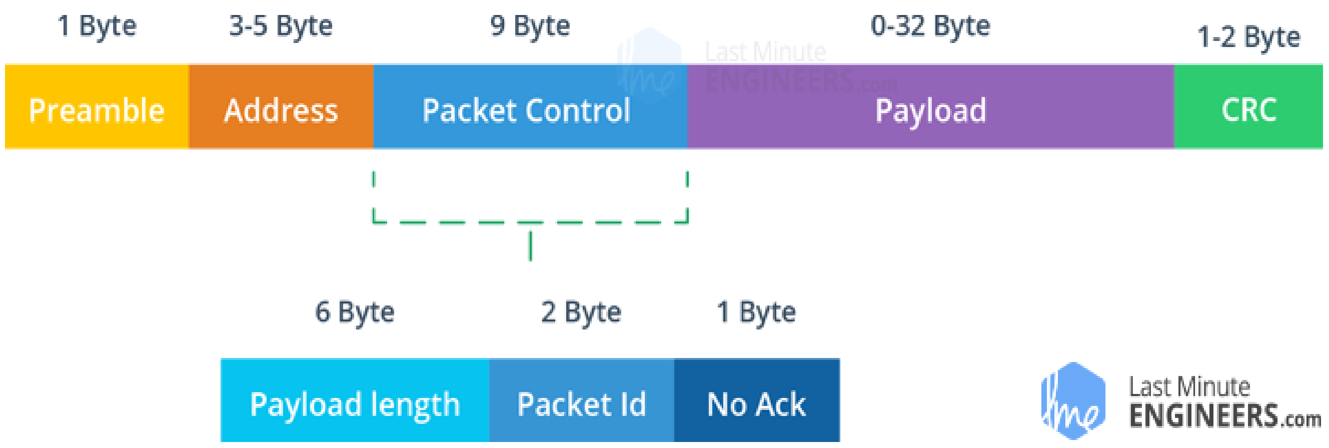


nRF24L01+ Multiceiver Network – Multiple Transmitters Single Receiver

To simplify the above diagram, imagine the primary receiver acting as a hub receiver collecting information from 6 different transmitter nodes simultaneously. The hub receiver can stop listening any time and acts as a transmitter. But this can only be done one pipe/node at a time.

Enhanced ShockBurst Protocol

The nRF24L01+ transceiver module uses a packet structure known as Enhanced ShockBurst. This simple packet structure is broken down into 5 different fields, which is illustrated below.



nRF24L01+ Enhanced ShockBurst Packet Structure

The original ShockBurst structure consisted only of Preamble, Address, Payload and the Cyclic Redundancy Check (CRC) fields. Enhanced ShockBurst brought about greater functionality for more enhanced communications using a newly introduced Packet Control Field (PCF).

This new structure is great for a number of reasons. Firstly, it allows for variable length payloads with a payload length specifier, meaning payloads can vary from 1 to 32 bytes.

Secondly, it provides each sent packet with a packet ID, which allows the receiving device to determine whether a message is new or whether it has been retransmitted (and thus can be ignored).

Finally, and most importantly, each message can request an acknowledgement to be sent when it is received by another device.

nRF24L01+ Automatic Packet Handling

Now, let’s discuss three scenarios to get a better understanding of how two nRF24L01+ modules transact with each other.

Transaction with acknowledgement and interruptThis is an example of positive scenario. Here the transmitter starts a communication by sending a data packet to the receiver. Once the whole packet is transmitted, it waits (around 130 µs) for the acknowledgement packet (ACK packet) to receive. When the receiver receives the packet, it sends ACK packet to the transmitter. On receiving the ACK packet the transmitter asserts interrupt (IRQ) signal to indicate the new data is available.

Last Minute ENGINEERS.com
NRF Tx
NRF Rx
Data Sent

Transaction with data packet lostThis is a negative scenario where a retransmission is needed due to loss of the packet transmitted. After the packet is transmitted, the transmitter waits for the ACK packet to receive. If the transmitter doesn’t get it within Auto-Retransmit-Delay (ARD) time, the packet is retransmitted. When the retransmitted packet is received by the receiver, the ACK packet is transmitted which in turn generates interrupt

Last Minute ENGINEERS.com
NRF Tx
NRF Rx
Data Sent

at the transmitter.

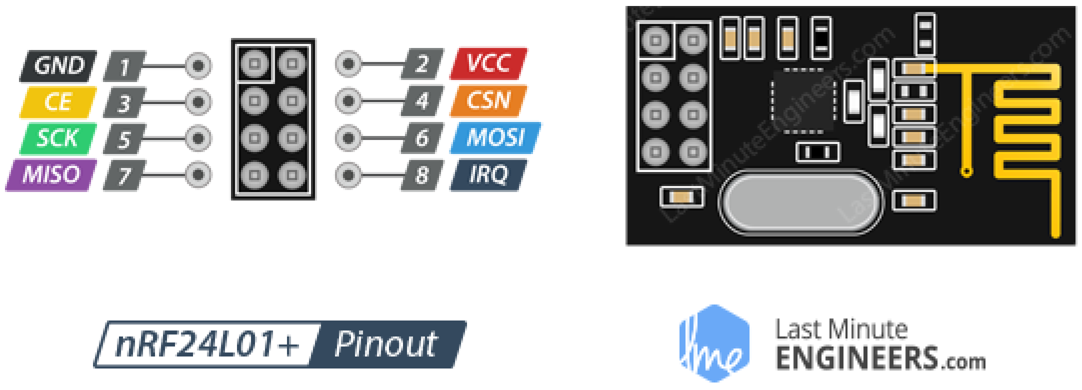
Transaction with acknowledgement lostThis is again a negative scenario where a retransmission is needed due to loss of the ACK packet. Here even if the receiver receives the packet in the first attempt, due to the loss of ACK packet, transmitter thinks the receiver has not got the packet at all. So, after the Auto-Retransmit-Delay time is over, it retransmits the packet. Now when receiver receives the packet containing same packet ID as previous, it discards it and sends ACK packet again.

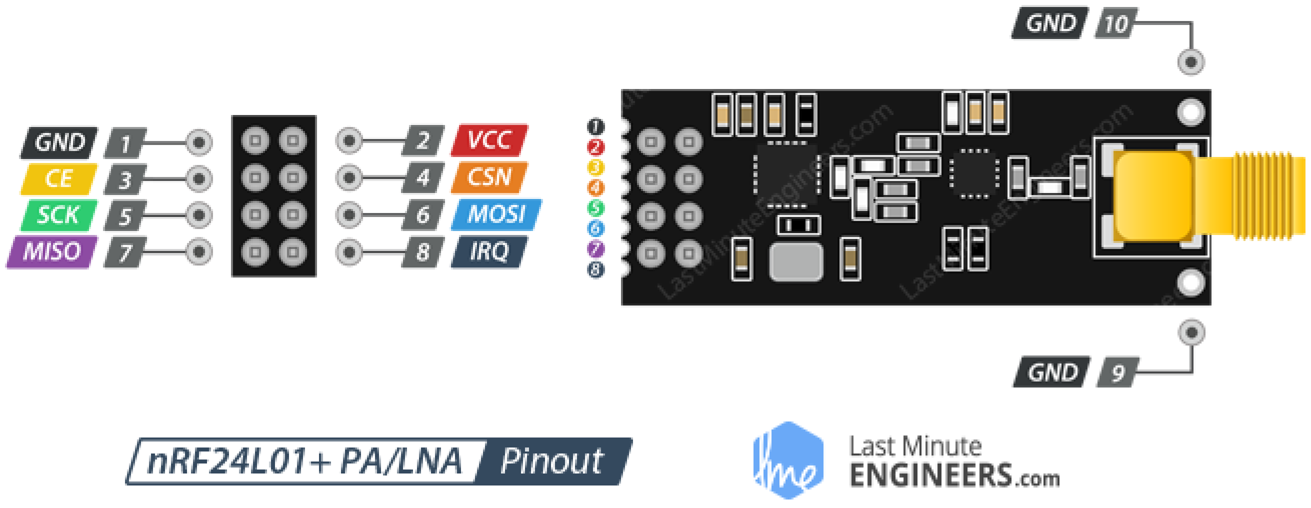
NRF Tx
NRF Rx
Data Sent
Last Minute ENGINEERS.com

This whole packet handling is done automatically by the nRF24L01+ chip without involvement of the microcontroller.

nRF24L01+ Transceiver Module Pinout

Let’s have a look at the pinout of both the versions of nRF24L01+ transceiver Module.





GND is the Ground Pin. It is usually marked by encasing the pin in a square so it can be used as a reference for identifying the other pins.
VCC supplies power for the module. This can be anywhere from 1.9 to 3.9 volts. You can connect it to 3.3V output from your Arduino. Remember connecting it to 5V pin will likely destroy your nRF24L01+ module!
CE (Chip Enable) is an active-HIGH pin. When selected the nRF24L01 will either transmit or receive, depending upon which mode it is currently in.
CSN (Chip Select Not) is an active-LOW pin and is normally kept HIGH. When this pin goes low, the nRF24L01 begins listening on its SPI port for data and processes it accordingly.
SCK (Serial Clock) accepts clock pulses provided by the SPI bus Master.
MOSI (Master Out Slave In) is SPI input to the nRF24L01.
MISO (Master In Slave Out) is SPI output from the nRF24L01.
IRQ is an interrupt pin that can alert the master when new data is available to process.

CHAPTER 5

FLEX SENSOR

5.1 FLEX SENSOR DESCRIPTON

A flex sensor is a kind of sensor which is used to measure the amount of defection otherwise bending. The designing of this sensor can be done by using materials like plastic and carbon. The carbon surface is arranged on a plastic strip as this strip is turned aside then the sensor's resistance will be changed. Thus, it is also named a bend sensor. As its varying resistance can be directly proportional to the quantity of turn thus it can also be employed like a goniometer.

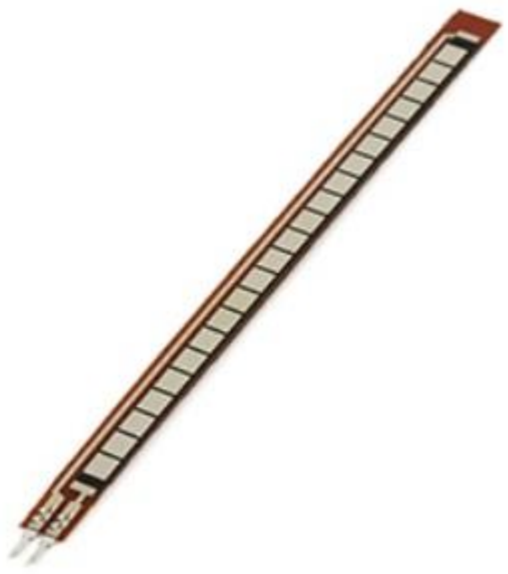


Fig 5.1 FLEX Sensor

5.2 FLEX SENSOR WORKING

This sensor works on the bending strip principle which means whenever the strip is twisted then its resistance will be changed. This can be measured with the help of any controller.

This sensor works similar to a variable resistance because when it twists then the resistance will be changed. The resistance change can depend on the linearity of the surface because the resistance will be dissimilar when it is level.

When the sensor is twisted 450 then the resistance would be dissimilar. Similarly, when this senor is twisted to 900 then the resistance would be dissimilar. These three are the flex sensor’s bending conditions.

According to these three cases, the resistance will be normal in the first case, the resistance will be double as contrasted with the first case, and the resistance will be four-time when compared with the first case. So the resistance will be increased when the angle is increased.

CHAPTER 6

SERVO MOTOR

A **servo motor** is an electrical device which can push or rotate an object with great precision. If you want to rotate and object at some specific angles or distance, then you use servo motor. It is just made up of simple motor which run through **servo mechanism**. If motor is used is DC powered then it is called DC servo motor, and if it is AC powered motor then it is called AC servo motor. We can get a very high torque servo motor in a small and light weight packages. Doe to these features they are being used in many applications like toy car, RC helicopters and planes, Robotics, Machine etc.

Servo motors are rated in kg/cm (kilogram per centimeter) most hobby servo motors are rated at 3kg/cm or 6kg/cm or 12kg/cm. This kg/cm tells you how much weight your servo motor can lift at a particular distance. For example: A 6kg/cm Servo motor should be able to lift 6kg if the load is suspended 1cm away from the motors shaft, the greater the distance the lesser the weight carrying capacity.

The position of a servo motor is decided by electrical pulse and its circuitry is placed beside the motor.

Servo Mechanism

It consists of three parts:

Controlled device

Output sensor

Feedback system

It is a closed loop system where it uses positive feedback system to control motion and final position of the shaft. Here the device is controlled by a feedback signal generated by comparing output signal and reference input signal.

Here reference input signal is compared to reference output signal and the third signal is produces by feedback system. And this third signal acts as input signal to control device. This signal is present as long as feedback signal is generated or there is difference between reference input signal and reference output signal. So the main task of servomechanism is to maintain output of a system at desired value at presence of noises.

Working principle of Servo Motors

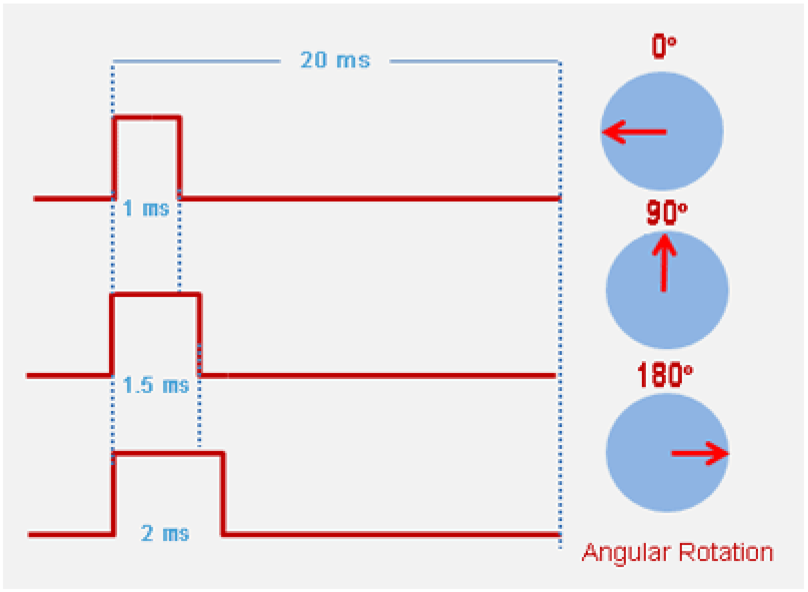
A servo consists of a Motor (DC or AC), a potentiometer, gear assembly and a controlling circuit. First of all we use gear assembly to reduce RPM and to increase torque of motor. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. Now an electrical signal is given to another input terminal of the error detector amplifier. Now difference between these two signals, one comes from potentiometer and another comes from other source, will be processed in feedback mechanism and output will be provided in term of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor shaft is connected with potentiometer and as motor rotates so the potentiometer and it will generate a signal. So as the potentiometer's angular position changes, its output feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stops rotating.

Controlling Servo Motor:

All motors have three wires coming out of them. Out of which two will be used for Supply (positive and negative) and one will be used for the signal that is to be sent from the MCU.

Servo motor is controlled by PWM (Pulse with Modulation) which is provided by the control wires. There is a minimum pulse, a maximum pulse and a repetition rate. Servo motor can turn 90 degree from either direction form its neutral position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position, such as if pulse is shorter than 1.5ms shaft moves to 0° and if it is longer than 1.5ms than it will turn the servo to 180°.

Servo motor works on **PWM (Pulse width modulation)** principle, means its angle of rotation is controlled by the duration of applied pulse to its Control PIN. Basically servo motor is made up of **DC motor which is controlled by a variable resistor (potentiometer) and some gears**. High speed force of DC motor is converted into torque by Gears. We know that WORK= FORCE X DISTANCE, in DC motor Force is less and distance (speed) is high and in Servo, force is High and distance is less. Potentiometer is connected to the output shaft of the Servo, to calculate the angle and stop the DC motor on required angle.



CHAPTER 7

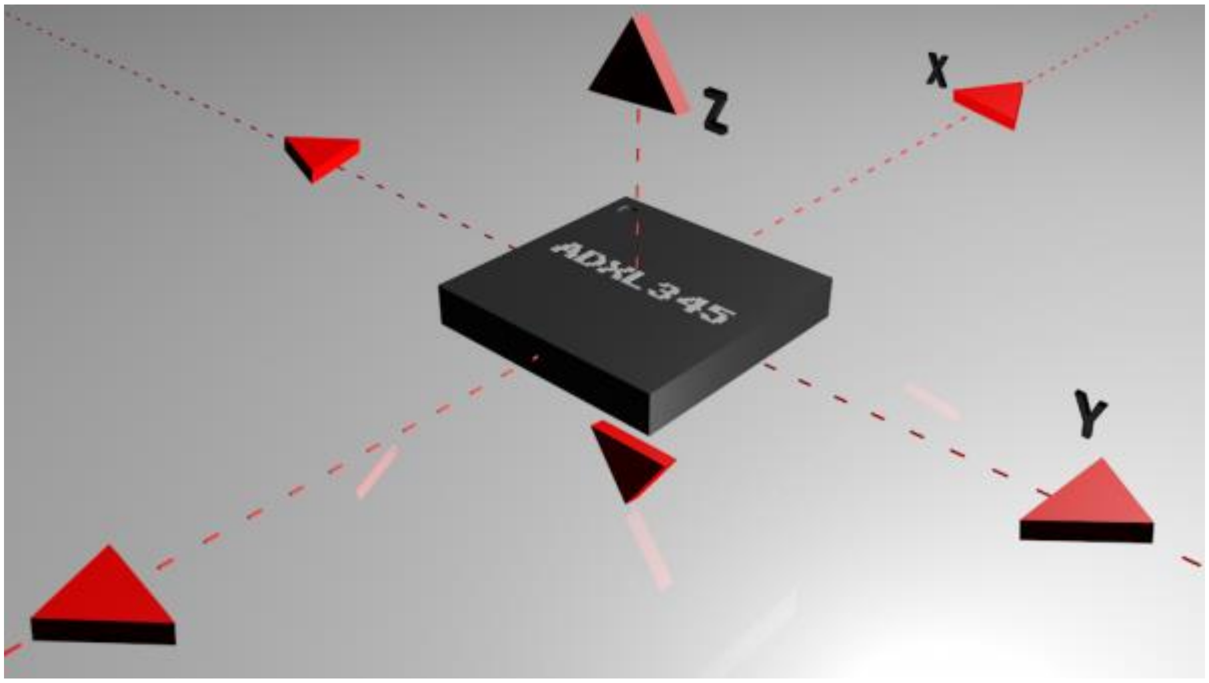
ACCELEROMETER

Accelerometers are devices that measure acceleration, which is the rate of change of the velocity of an object. They measure in meters per second squared (m/s^2) or in G-forces (g). A single G-force for us here on planet Earth is equivalent to 9.8 m/s^2, but this does vary slightly with elevation (and will be a different value on different planets due to variations in gravitational pull). Accelerometers are useful for sensing vibrations in systems or for orientation applications.

ADXL345 Breakout Board

WORKING

Accelerometers are electromechanical devices that sense either static or dynamic forces of acceleration. Static forces include gravity, while dynamic forces can include vibrations and movement.



Axes of measurement for a triple axis accelerometer

Accelerometers can measure acceleration on one, two, or three axes. 3-axis units are becoming more common as the cost of development for them decreases.

Generally, accelerometers contain capacitive plates internally. Some of these are fixed, while others are attached to minuscule springs that move internally as acceleration forces act upon the sensor. As these plates move in relation to each other, the capacitance between them changes. From these changes in capacitance, the acceleration can be determined.

Other accelerometers can be centered around piezoelectric materials. These tiny crystal structures output electrical charge when placed under mechanical stress (e.g. acceleration).

CHAPTER 8

PROGRAMMING

RECEIVER CODE

```
#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

#include <Servo.h>

RF24 radio(7, 8);

const byte rxAddr[] = "0xE8E8F0F0E1LL";

Servo finger1 ;

int servopin1 = 4;

int msg[1];

int data; //data variable

int pos; //position variable

void setup()

{

  while (!Serial);

  Serial.begin(9600);

  finger1.attach(servopin1);

  pinMode(servopin1, OUTPUT);

  radio.begin();

  radio.openReadingPipe(0, rxAddr);
  radio.startListening();

}

void loop()

{

  if (radio.available())

  {

     radio.read(&msg, sizeof(msg));

       if(msg[0] <21 && msg[0]>10){

      data = msg[0], pos=map(data, 11, 20, 0, 180);

      finger1.write(pos);

      Serial.println(pos);

    }

  }

}
```

8.2 TRANSMITTER CODE

```
#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

RF24 radio(7, 8);

const byte rxAddr[] = "0xE8E8F0F0E1LL";

int msg[1];

//define the flex sensor input pins
```

```
int flex_5 = A1;

int flex_4 = A4;

int flex_3 = A3;

int flex_2 = A2;

int flex_1 = A5;

//define variables for flex sensor values

int flex_5_val;

int flex_4_val;

int flex_3_val;

int flex_2_val;

int flex_1_val;

void setup()

{

  Serial.begin(9600);

  radio.begin();

  pinMode(flex_5 , INPUT);

  radio.setRetries(15, 15);

  radio.openWritingPipe(rxAddr);

  radio.stopListening();

}

void loop()

{
```

```
flex_5_val = analogRead(flex_5);

Serial.print(flex_5_val);                          //175 - 0

flex_5_val = map(flex_5_val, 60 , 140 , 11 , 20);

Serial.print(",\t");

flex_5_val = constrain(flex_5_val , 11 , 20 );

Serial.println(flex_5_val);

msg[0] = flex_5_val;

radio.write(  &msg, sizeof(msg) );

flex_4_val = analogRead(flex_4);

flex_4_val = map(flex_4_val, 1023, 0, 11, 20);

msg[0] = flex_4_val;

  }
```

CHAPTER 9

APPLICATIONS

The applications of robotic hands are numerous. They are listed below.

- Extra hand in chemical fields so that the skin diseases are avoided.
- Physically challenged people use it as an extra alternative hand.
- Two hands for the same operation can be utilized with this animatronic hand.
- Its applications in space for repair of space station are useful.
- Diffusion of bombs with these animatronic hands are preferred where there is high risk of lives.

RESULTS & DISCUSSION

A mobile robotic system has been developed which works according to your hand gesture. It provides a better way to control a robotic arm using accelerometer which is more intuitive and easy to work. The RF module is working on the frequency of 433 MHz and has a range of 50-80 meters. This robot can be used in industries to perform hazardous tasks. It can also be upgraded to bomb detecting robot as it has robotic arm it can also lift the bomb. GPS system can be added to the robot by the help of which its location can be tracked.

The gesture controlled robotic arm for industrial application is designed and implemented. The movement is precise, accurate, as well easy to control and friendly to use. The robotic arm has been made very carefully and in a detailed manner so that the movement of the robot can be controlled accurately. This robotic arm control method will be helpful in many aspects to make human life comfortable and easy.