School of Engineering & Applied Science

Embedded System Design Project

Security System For Warehouse

Group No. - 16

Dhara Vora - 1741046

Krushna Shah - 1741086

Hetvi Suthar - 1741089

Kruti Yadav - 1741096

Contents

- ❏ Details of actuators and Display devices

9. Details of supporting tools
   - ❏ Complete details of Android App. Or any other tools If used in project

10. Complete Program
    - ❏ Complete C Program

11. Summary
    - ❏ Summary of the working of the project with time-line

12. References
    - ❏ References

Appendix A Datasheets
- ❏ Selected Pages of Datasheets of all sensors and actuators used

Appendix B Programming Review
- ❏ Compare C programming with two other languages

Appendix C Troubleshooting
- ❏ Explain at least three examples of troubleshooting and debugging during the project work

Chapter 1 Introduction

Introduction:

The main concept of the project is to provide secure, intelligent, efficient and reliable parking system for the warehouse. In real life scenarios, many unwanted situations happen in the warehouses like unknown vehicle enters in the warehouse parking to park the vehicle or any unknown truck enters to steal the goods, etc. So as an efficient solution we have come up with an idea about how warehouse' system can be managed by using new technologies and methodologies.

Description:

Security system in warehouse project is mainly based on Radio Frequency Identification (RFID) technology. We are using RFID tags so that check-in and check-out process becomes significantly faster. The RFID reader is placed near the boom barrier to identify the vehicle by tracking tag attached to the
vehicle. After that, we are using IR sensor to detect the vehicle. According to the arrival time of the vehicle we will send tag-id and in-time to the google sheet using Node MCU. If any unknown vehicle tries to enter, then using GSM (Global System for Mobile) message will be sent to an owner of warehouse that "Unknown vehicle tried to enter!" Now, when the vehicle passes the boom barrier and if there is a space in parking, then the LED \ LCD
display will indicate the full or empty parking slot - bright led will indicate the empty parking slot and dim led will indicate the full parking slot. This indication is based on the the detection of the IR sensor. At exit, there is one another RFID reader. That will scan the RFID tag, and according to the tag-id we will send out-time to the google sheet.

Motivation:

The aim of development of this system is to basically reduce human interference and apply smart technologies which are related to automation field to our day to day applications and make human lives more comfortable. Creating an automated system which not only helps users to make parking much more efficient and faster but also automates the payment gateway using RFID thus saving the user a lot of time and the owner of the warehouse will get all the details about check-in and check-out time of the particular vehicle.By doing this project, we will also learn some new modules and concepts like send data to google drive using Wi-fi Module (NodeMCU) and also GSM module.

Chapter 2 Market Survey

Market Survey of current Products:

Existing System:

The existing parking management systems for warehouse require human efforts, and manual recording in excel sheets and on paper. For huge parking, scenario is very hectic to keep track of.And other thing is there is no security plan. This system have very high chances for entering wrong/unknown vehicle.

Our project idea for more efficient way of security :

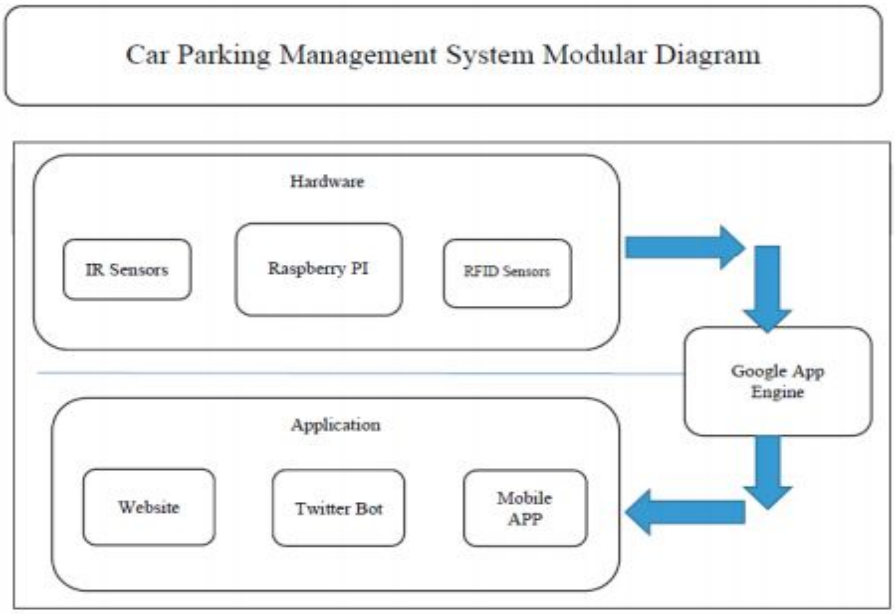
In our project we are using sensors along with the RFID, Wi-fi module and GPS module. RFID tag cannot be cloned, so cannot be cheated.RFID reader will detect the tag and if it is correct RFID tag, then the gate will automatically open for parking. And if the RFID reader will detect the wrong RFID tag, then the gate will remain close and automatically the owner of the warehouse will get a message that 'some unknown vehicle has tried to enter.' And also using the WI-fi module, we will store the in-time and out-time of the each vehicle in the datasheet. IR sensors will detects the vehicle at the parking slot and according to that, at the entrance LCD display will display the free parking slot.

Comparison with the research paper :

For this project, we have referred one research paper published by Assistant Professor of School of Computing Science and Engineering, VIT University, Chennai. Their main objective is to avoid the cramming in the car parking area by implementing an efficient car parking system along with a user-friendly application for an ease of use. Normally at public places such as multiplex theatres, market areas, hospitals, function-halls, offices

and shopping malls, one experiences the discomfort in looking out for a vacant parking slot, though it's a paid facility with an attendant/ security guard. The parking management system is proposed to demonstrate hazel free parking. The proposed system uses infrared transmitter-receiver pairs that remotely communicate the status of parking occupancy to the raspberry pi and displays the vacant slots on the display at the entrance of the parking so that the user gets to know the availability /unavailability of parking space prior to his/her entry into the parking place. Implementation involves minimal human interaction and provides a seamless parking experience thereby reducing a lot of time wasted by the user in parking his/her vehicle.

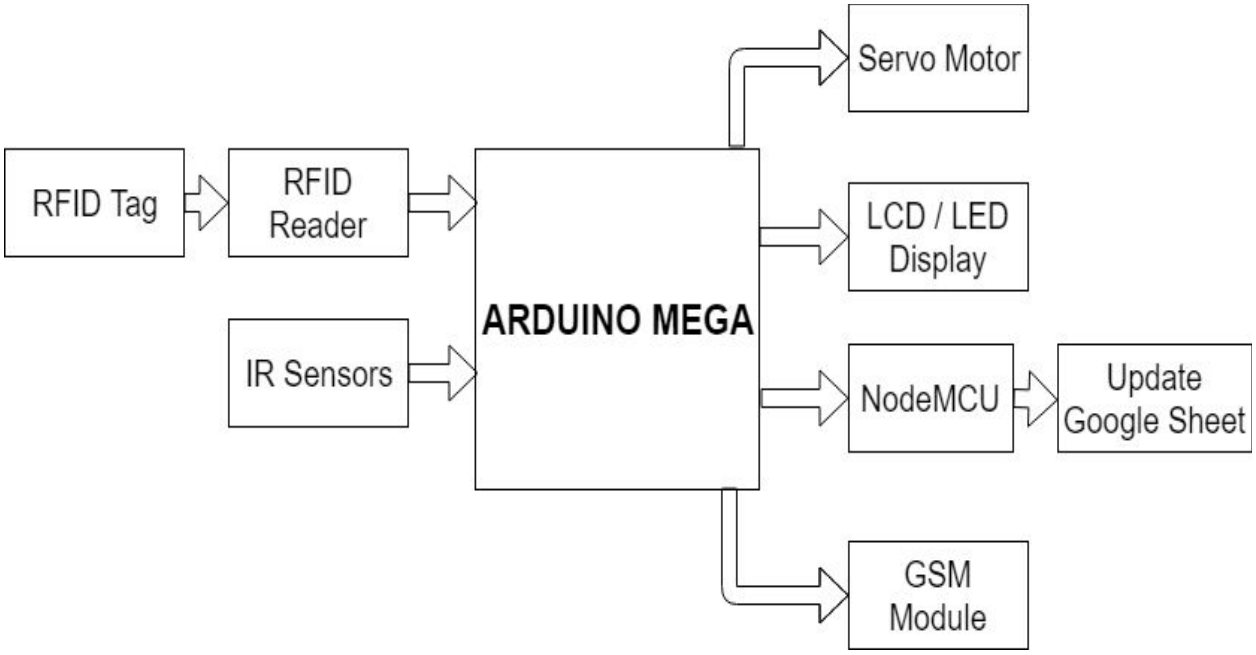
The Overall Engineering Design is described in the below image. The hardware comprises of IR Sensors, Raspberry pi and RFID Sensors which communicates with the Google App Engine and is rendered on various Applications such as the Website, Twitter bot and the Mobile Application. This detailed architecture explains about the entire working system of the efficient car parking model with various personalized features. The sensor used in this project is an infrared sensor which determines whether the slot is occupied or unoccupied. These sensors are connected to the raspberry pi. The output of these sensors is sent to the database through the raspberry pi . Once the database is updated the result is displayed on the video monitor at the entrances of the parking level. This result is displayed using the website url which is updated every 5 seconds. The other module of CPMS is the payment gateway using RFID. There will be an RFID Tag attached to every vehicle. As soon as the vehicle passes through the entry gate the RFID Reader reads the tag and gets the Unique ID and then logs an entry into the database and upon exit another RFID Reader reads the tag and deducts money from the users account based on the time spent in the parking lot and hence completes the transaction.



Their main aim is to design an integrated system which involves two components namely Parking Allocation and Seamless Parking. The Parking Allocation component consists of sensors in front each slot and when a vehicle enters into the slot, the database is updated and the changes are reflected immediately on the nearby display. The Seamless parking component consists of a RFID Tag attached to the windshield of vehicle. When the car is passed through the entrance the RFID Scanner scans the tag stores the timestamp of the entry and at the exit the tag is read again and the total time is calculated and reflected in the users account, thus saving the hassle of human intervention and saving an ample amount of time. We have interfaced 6 IR Sensors and an RFID Reader module (EM-18) using a raspberry pi. The IR senses the presence of a vehicle in the parking slot and updates the database. The RFID is used for identification and transaction. We have made an android application, twitter bot, telegram bot, website to further simplify the process of getting data about the parking slot availability and effort less parking.

Chapter 3 Block Diagram

Block Diagram :



Chapter 4 Circuit Diagram

List of Components:

Arduino Mega
RFID Tag & Reader
IR Sensor
Servo Motor
LED
LCD Display
Node MCU - ESP8266 Wifi Module
GSM Module

Selection Criteria of Components:

RFID Tag & Reader :

RFID tags and readers are used in this project so that we can detect the vehicle which has valid RFID tag via RFID reader. Using this we can prohibit to enter any unknown vehicle at the warehouse' parking.

IR Sensors :

We have selected IR sensors so that it can detects the vehicle at the parking slot that whether the parking slot is empty or not.

Servo Motor :

We have selected servo motor to control the boom barrier. Whenever the vehicle enters which has valid RFID, using this servo motor boom barrier will open the gate.

LEDs :

We have selected LEDs to indicate whether the parking slot is empty or not. Bright LED will indicate the empty parking slot and dim LED will indicate the empty parking slot.

LCD Display :

We have selected LCD display to display the empty or full parking slots.

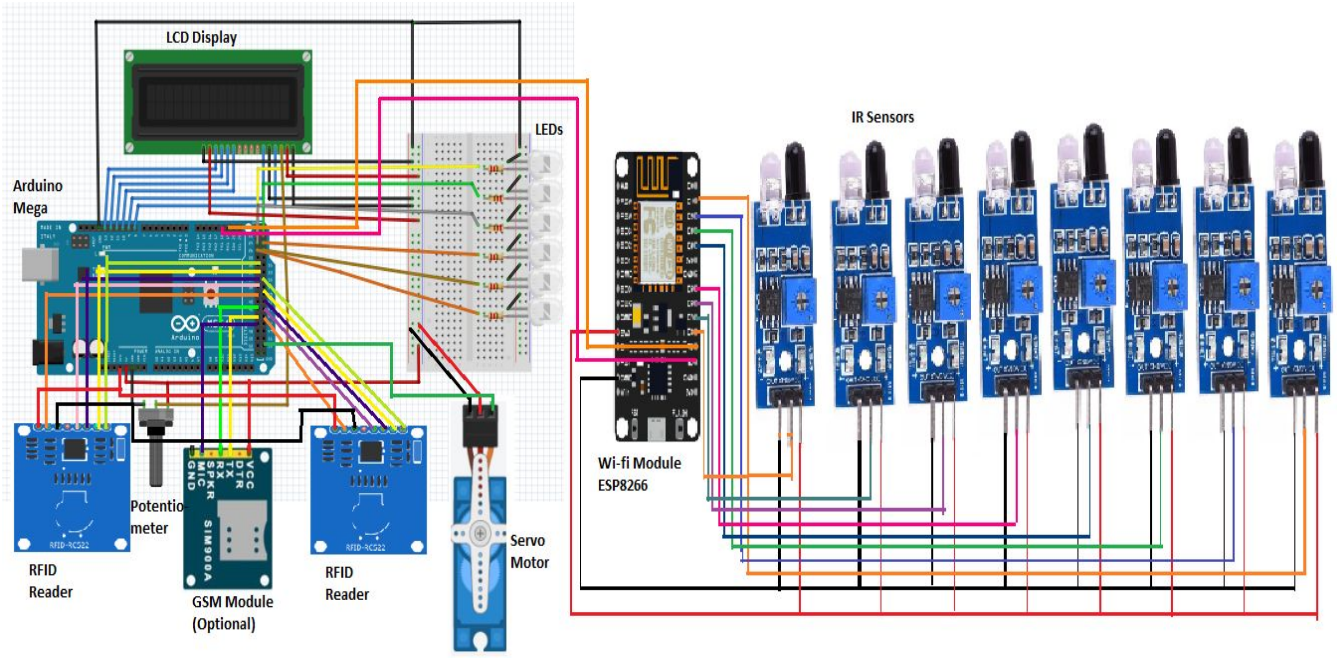
Wi-fi Module (NodeMCU - ESP8266) :

We have selected ESP8266 wifi module to store the vehicle's RFID tag id, in-time and out-time in the google sheet.

GSM Module : (Optional)

We have selected GSM module to send the message to the owner of the warehouse, whenever the unknown vehicle tries to enter in the warehouse.

Circuit Diagram :



Chapter 5 Arduino Features

Details of Arduino and compare with other platforms:

➢Arduino Mega :

Specifications:

- Microcontroller ATmega1280
- Operating Voltage : 5V
- Input Voltage (recommended) : 7-12V
- Input Voltage (limits) : 6-20V
- Digital I/O Pins : 54 (of which 15 provide PWM output)
- Analog Input Pins : 16
- DC Current for 3.3V Pin : 50 mA
- Flash Memory : 128 KB of which 4 KB used by bootloader
- SRAM : 8 KB
- EEPROM : 4 KB

- Clock Speed : 16 MHz

➢Raspberry Pi :

RASPBERRYPI-MODB-1GB

RPI-MODB-16GB-NOOBS

Specification:

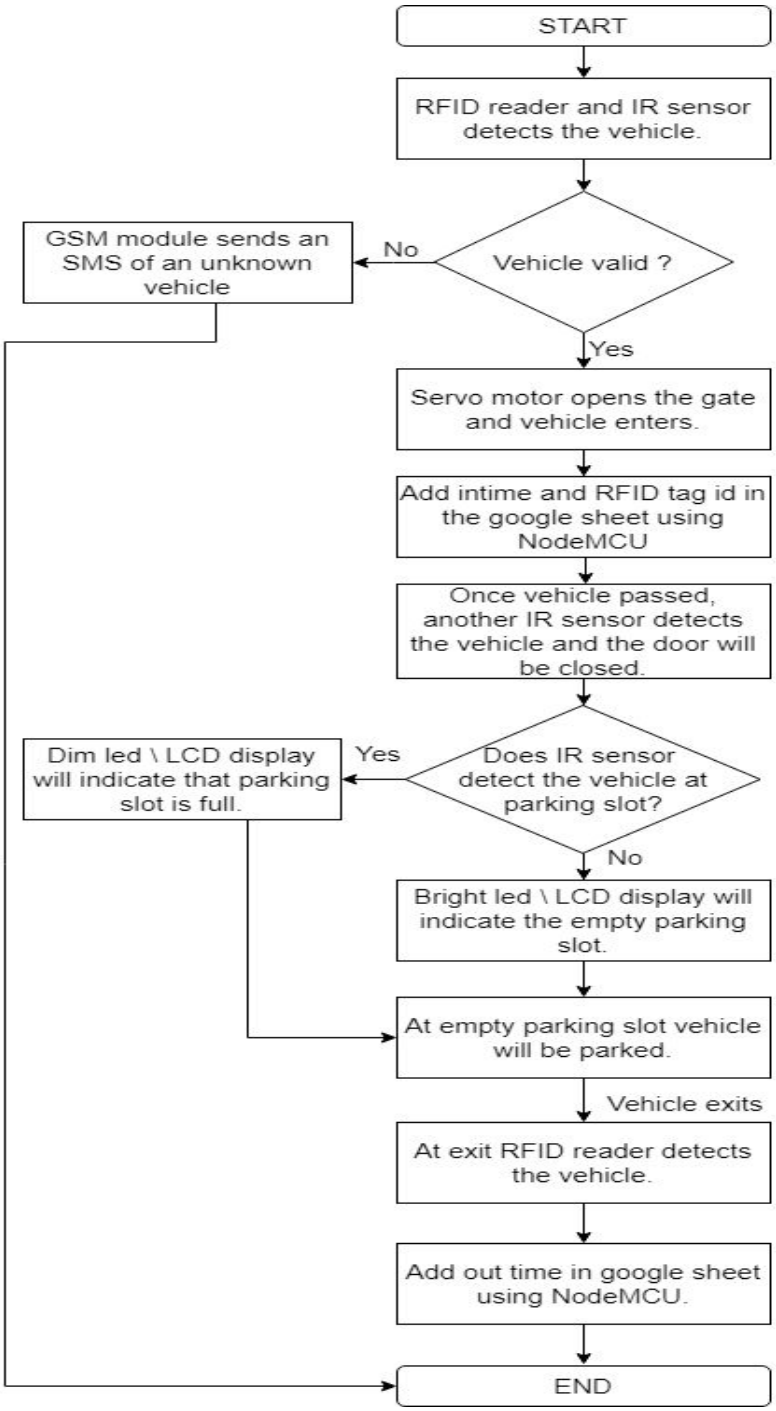
- Broadcom BCM2837 64bit ARMv7 Quad Core Processor powered Single Board Computer running at 1.2GHz
- 1GB RAM
- BCM43143 WiFi on board
- Bluetooth Low Energy (BLE) on board
- 40pin extended GPIO
- 4 x USB 2 ports
- 4 pole Stereo output and Composite video port
- Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display Micro SD port for loading your operating system and storing data Upgraded switched Micro USB power source (now supports up to 2.4 Amps)

- Expected to have the same form factor has the Pi 2 Model B, however the LEDs will change position

	Arduino Uno	Raspberry Pi Model B+	Intel Edison
Price	$30	$35	$50 (board not included)
Size	7.6 x 1.9 x 6.4 cm	8.5 x 5.6 x 1.7 cm	3.55 x 2.5 x .39 cm
Memory	0.002MB	512MB	1 GB
GPIO	14	40	40
Clock Speed	16 MHz	700 MHz	500 MHz, 100 MHz
On Board Network	None	10/100 BaseT Ethernet socket	Dual-band (2.4 and 5 GHz) Wifi, Bluetooth 4.0
Multitasking	No	Yes	Yes
Input voltage	7 to 12 V	5 V	3.3 to 4.5 V
Flash memory	32KB	Micro SD card	4 GB eMMC
USB	One, input only	Four, peripherals OK	One, peripherals OK
Operating System	None	Linux distributions	Yocto Linux v1.6
Integrated Development Environment	Arduino IDE	Scratch, IDLE, anything with Linux support	Arduino IDE, Eclipse, Intel XDK

Chapter 6 Program Flow Chart

Flow chart of the program :



Chapter 7 Sensors

Details of sensors :

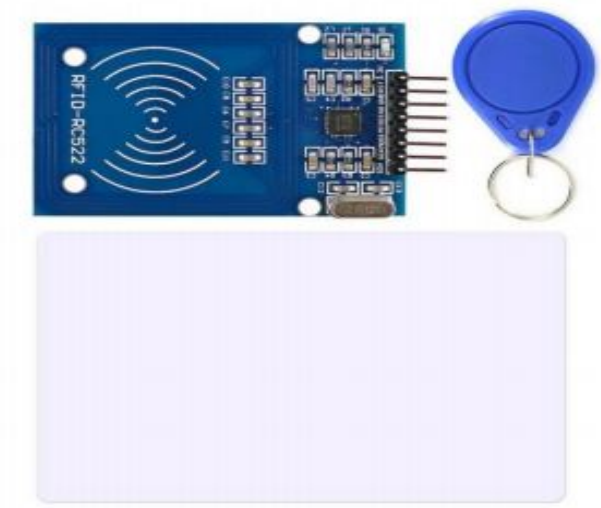
1) RFID Tag and RFID Reader :

RFID means radio-frequency identification. RFID uses electromagnetic fields to transfer data over short distance.

An RFID system uses:

- Two way radio transmitter-receiver, the **reader**, that send a signal to the tag and read its response.
- **Tag**s attached to the object to be identified, in this example we have a keychain and an electromagnetic card. Each tag has his own identification(UID).

The **RFID reader** is one kind of wireless module used for transferring the data to identify and track tags which are connected to objects. The RFID tag mainly includes the stored information. Some of the RFID tags are run by electromagnetic induction from magnetic fields formed nearby the reader. RFID reader comprises an RF module that works as a transmitter as well as a receiver of RF (radio frequency) signals.



The TX of the RF module is inbuilt with an oscillator to make the carrier frequency. A modulator to intrude commands upon this carrier signal and an amplifier to raise the signal sample to wake the tag. The RX (receiver) of the RFID module contains a demodulator to remove the returned information and also grips an amplifier for supporting the signal of processing. A microprocessor is used for forming the control unit, which uses an operating system, a memory of the module filter and also stores the data.

An **RFID tag**, or transponder, consists of a chip and an antenna. [7]A chip can store a unique serial number or other information based on the tag's type of memory, which can be read-only, read-write, or write once read-many (WORM). The antenna, which is attached to the microchip, transmits information from the chip to the reader. Typically, a larger antenna indicates a longer read range. The tag is attached to or embedded in an object to be identified.

➢ **MFRC522 RFID Module :**

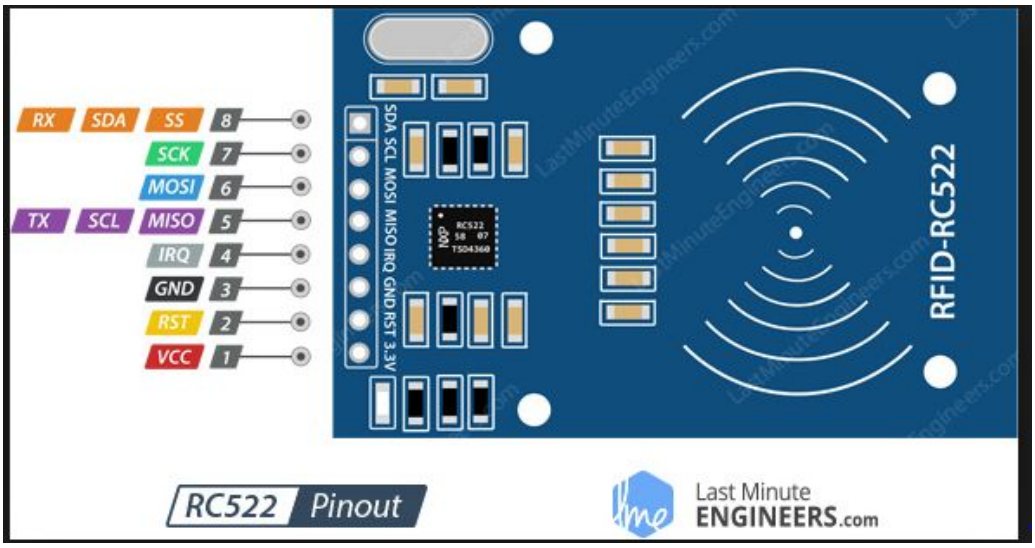
Mifare RC522 is the high integrated RFID card reader which works on non-contact 13.56 MHz communication, is designed by NXP as low power consumption, low cost and compact size read and write chip, is the best choice in the development of smart meters and portable hand-held devices. MFRC522 use the advanced

modulation system, fully integrated at 13.56MHz with all kinds of positive non-contact communication protocols. Support 14443A compatible answer signal. DSP deal with ISO14443A frames and error correction. Furthermore, it also supports rapid CRYPTO1 encryption to validate Mifare series products. MFRC522 support Mifare series higher speed non-contact communication, duplex communication speed up to 424 kb/s. As a new family member in 13.56MHz RFID family, MFRC522 has many similarities to MF RC5200 and MFRC530, and also has more new features. This module can fit directly in handheld devices for mass production. Module use 3.3V power supply, and can communicate directly with any CPU board by connecting through SPI protocol, which ensure reliable work, good reading distance.

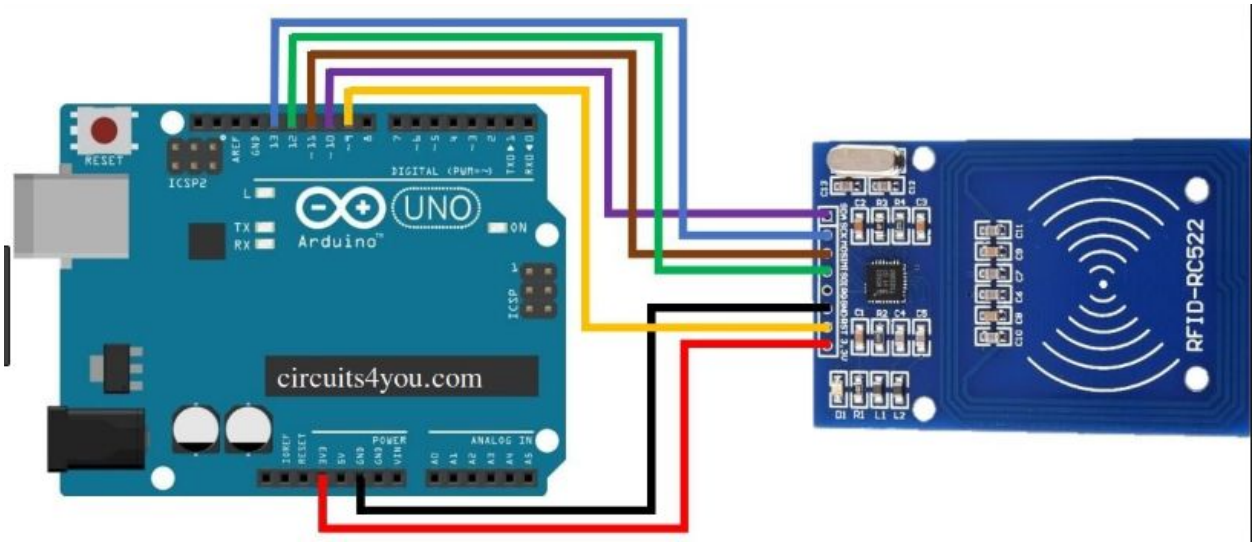
➢ **Specifications :**

- Module Name:MF522-ED
- Working current：13 - 26mA / DC 3.3V
- Standby current：10 - 13mA / DC 3.3V
- Sleep current：<80uA
- Peak current：<30mA
- Working frequency：13.56MHz
- Card reading distance ：0～60mm （Mifare1 card）
- Protocol：SPI
- Data communication speed：10Mbit/s Max.
- Card types supported: Mifare1 S50, Mifare1 S70, Mifare UltraLight, Mifare Pro, Mifare Desfire
- Dimension：40mm × 60mm
- Working temperature：-20—80 degree
- Storage temperature：-40—85 degree
- Humidity：relevant humidity 5%—95%

➢ Pin Description:



➢ Interfacing with Arduino:



➢ C Code:

```
#define RST_PIN         9          // Configurable, see typical pin layout above
#define SS_PIN          10         // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance

void setup() {
  Serial.begin(9600);   // Initialize serial communications with the PC
  while (!Serial);    // Do nothing if no serial port is opened (added for Arduinos based on ATMEGA32U4)
  SPI.begin();      // Init SPI bus
  mfrc522.PCD_Init();   // Init MFRC522
  mfrc522.PCD_DumpVersionToSerial();  // Show details of PCD - MFRC522 Card Reader details
  Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks..."));
}

void loop() {
  // Look for new cards
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }

  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  // Dump debug info about the card; PICC_HaltA() is automatically called
  mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
  Serial.println(millis());
}
```

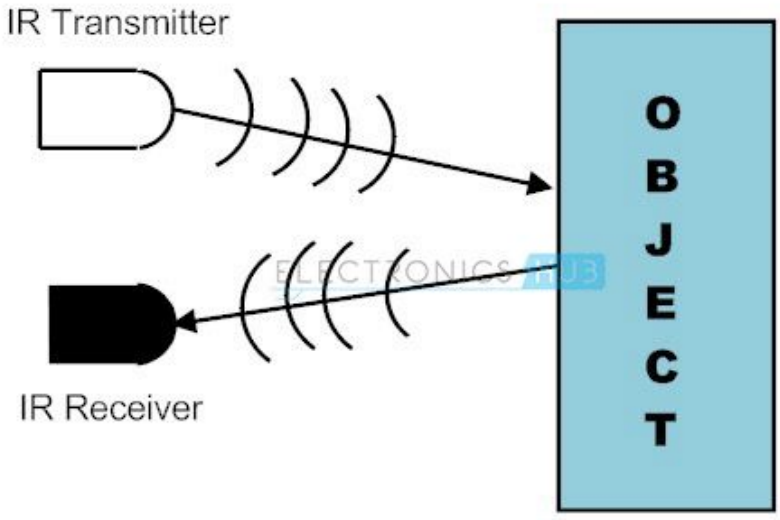
```
Done Saving.
Sketch uses 8748 bytes (3%) of program storage space. Maximum is 253952 bytes.
Global variables use 283 bytes (3%) of dynamic memory, leaving 7909 bytes for local variables. Maximum is 8192 bytes.
```

Size of code : 8748 bytes
Execution Time : 1039 ms

2) IR Sensor :

An **infrared sensor(IR)** is an electronic device, that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion.These types of sensors measures only infrared radiation, rather than emitting it that is called as a passive IR sensor. Usually in the infrared spectrum, all the objects radiate some form of thermal radiations. These types of radiations are invisible to our eyes, that can be detected by an infrared sensor.The emitter is simply an IR LED (Light Emitting Diode)(transmitter) and the detector is simply an IR photodiode(receiver) which is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode,

The resistances and these output voltages, change in proportion to the magnitude of the IR light received.



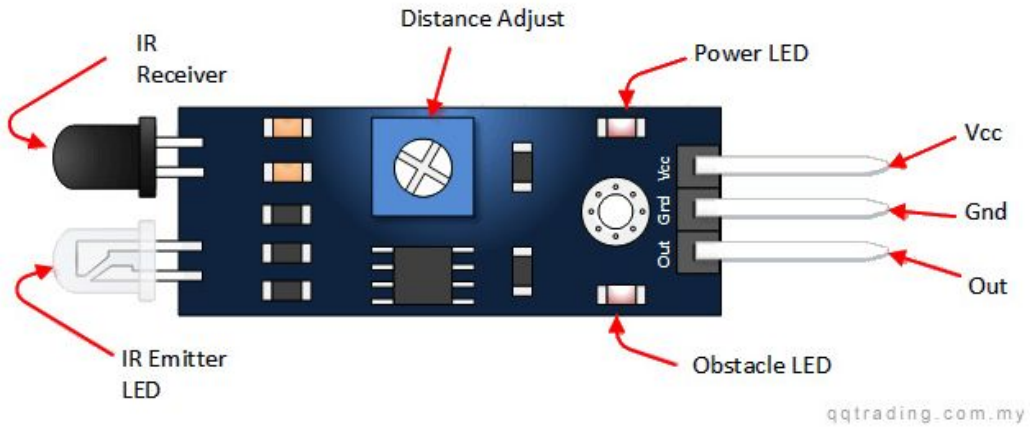
➢ **Features:**

- There is an obstacle, the green indicator light on the circuit board
- Digital output signal
- Detection distance: 2 ~ 30cm
- Detection angle: 35 ° Degree
- Comparator chip: LM393
- Adjustable detection distance range via potentiometer:
    - Clockwise: Increase detection distance
    - Counter-clockwise: Reduce detection distance

➢ **Specifications:**

- Working voltage: 3 - 5V DC
- Output type: Digital switching output (0 and 1)
- 3mm screw holes for easy mounting
- Board size: 3.2 x 1.4cm

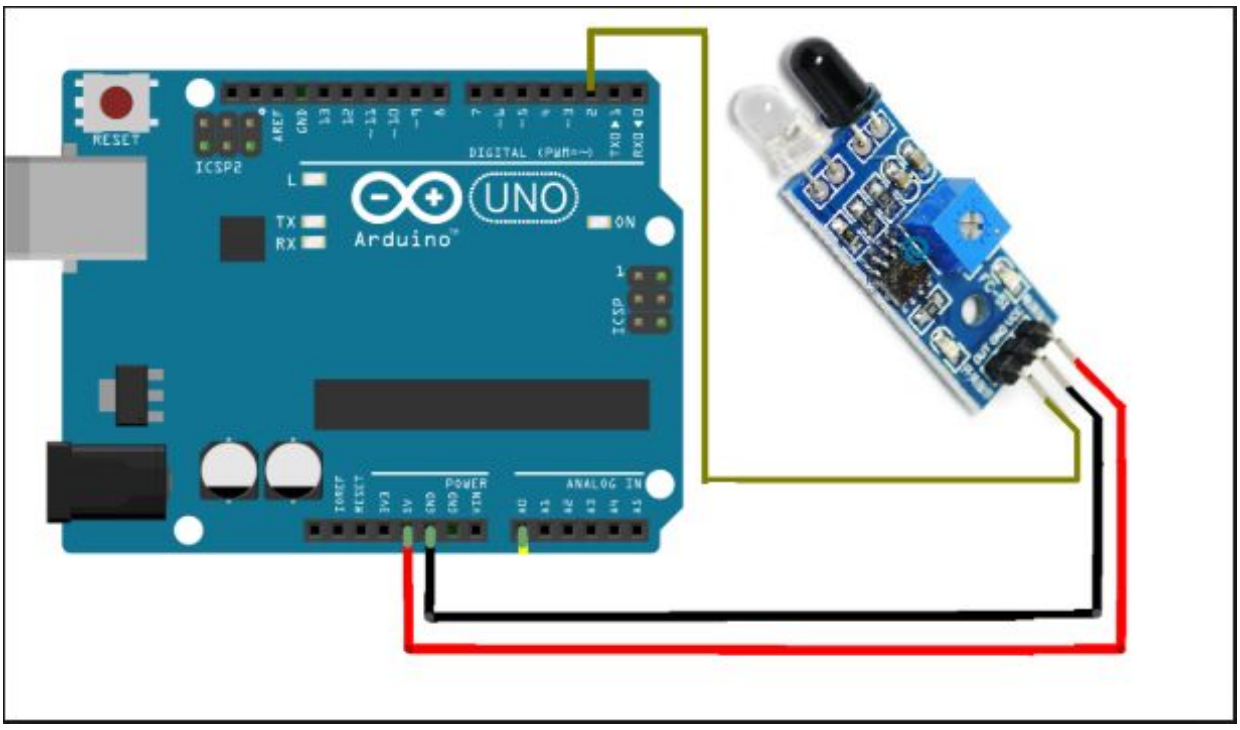
➢ **Pin Description:**



➢ **Pin,Control Indicator**	**Description**
Vcc	3.3 to 5V dc Supply Input
Gnd	Ground Input
Out	Output that goes low when obstacle is in range
Power LED	Illuminates when power is applied
Obstacle LED	Illuminates when obstacle is detected
Distance Adjust	Adjust detection distance. CCW decreases distance. CW increases distance.
IR Emitter	Infrared emitter LED

IR Receiver	Infrared receiver that receives signal transmitted by Infrared emitter.

➢ **Interfacing with Arduino:**



➢ **C Code:**

```
int LED = 13; // Use the onboard Uno LED
int obstaclePin = 7;  // This is our input pin
int hasObstacle = HIGH;  // HIGH MEANS NO OBSTACLE

void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
  pinMode(obstaclePin, INPUT);
  Serial.begin(9600);
}
void loop() {
  hasObstacle = digitalRead(obstaclePin); //Reads the output of the obstacle sensor from the 7th PIN of the Digital section of the arduino
  if (hasObstacle == LOW) //LOW means something is ahead, so illuminates the 13th Port connected LED
  {
    Serial.println("Stop something is ahead!!");
    digitalWrite(LED, HIGH);//Illuminates the 13th Port LED
  }
  else
  {
    Serial.println("Path is clear");
    digitalWrite(LED, LOW);
  }
  delay(200);
  Serial.println(millis());
}
```

```
Done uploading.

Sketch uses 2962 bytes (1%) of program storage space. Maximum is 253952 bytes.
Global variables use 230 bytes (2%) of dynamic memory, leaving 7962 bytes for local variables. Maximum is 8192 bytes.
```

Size of code : 2962 bytes

Execution Time : 199 ms

Chapter 8 Actuators and Displays

Details of actuators :

1) Servo Motor :

Servo motor used to adjust the speed control at high torques and accurate positioning. Parts required are motor position sensor and a highly developed controller. These motors can be categorized according the servo motor controlled by servomechanism. If DC motor is controlled using this mechanism, then it is named as a DC servo motor. Servo motors are available in power ratings from fraction of a watt to 100 watts.The rotor of a servo motor is designed longer in length and smaller in diameter so that it has low inertia.

➢Features :

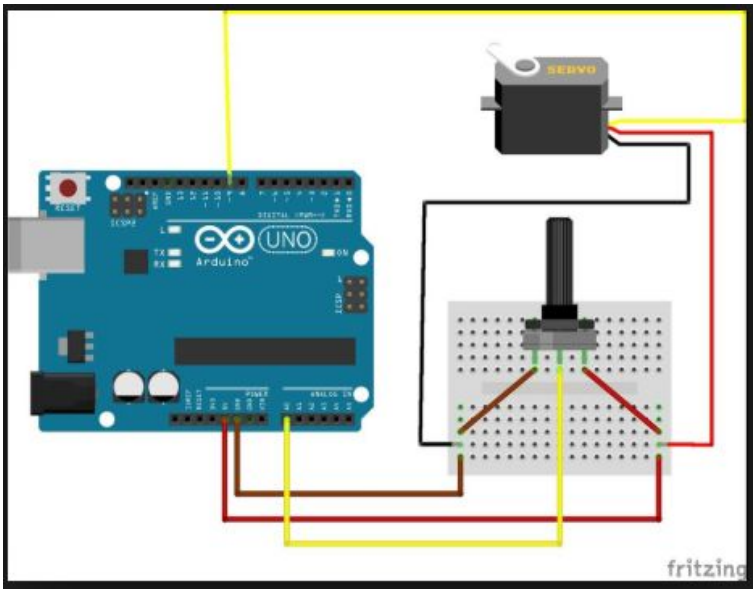
- Operating Voltage is +5V typically
- Torque: 2.5kg/cm
- Operating speed is 0.1s/60°

- Gear Type: Plastic
- Rotation : 0°-180°
- Weight of motor : 9gm
- Package includes gear horns and screws

➢Pin Description :

Wire Number	Wire Colour	Description
1	Brown	Ground wire connected to the ground of system
2	Red	Powers the motor typically +5V is used
3	Orange	PWM signal is given in through this wire to drive the motor.

➢Interfacing with Arduino :



➢C Code :

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  Serial.begin(9600);
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  Serial.println(millis());
}
```

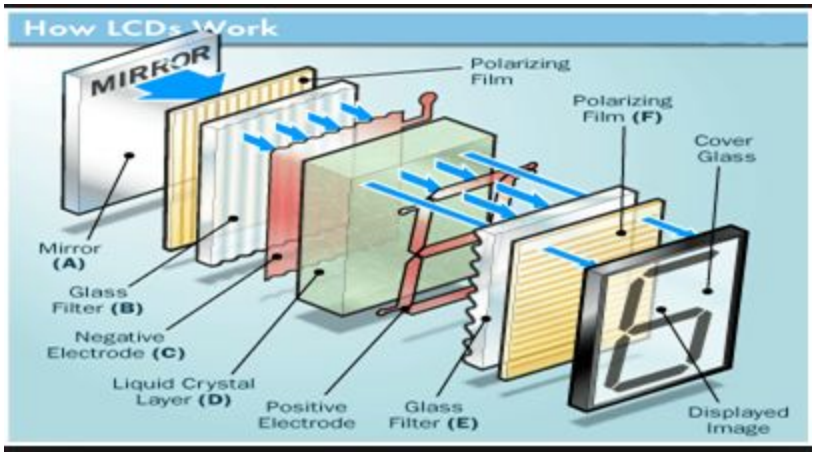
```
Done uploading.

Sketch uses 4326 bytes (1%) of program storage space. Maximum is 253952 bytes.
Global variables use 342 bytes (4%) of dynamic memory, leaving 7850 bytes for local variables. Maximum is 8192 bytes.
```

Size of Code : 4220 bytes
Execution Time : 5450 ms

Details of Display devices :

1) LCD Display :

Liquid crystal display is composed of several layers which include two polarized panel filters and electrodes. LCD technology is used for displaying the image in notebook or some other electronic devices like mini computers. Light is projected on a layer of liquid crystal. This combination of colored light with the grayscale image of the crystal (formed as electric current flows through the crystal) from the colored image. This image is displayed on the screen.

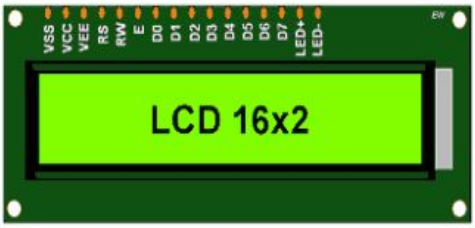


➢Features :

- Display Modes : STN , BLUB
- Display Formats : 16 characters x 2 lines
- Viewing Directions : 6 O’ clock
- Input Data : 4-bits or 8-bits interface available
- Display Font : 5 x 8 Dots

- Power Supply : 5V
- Driving Scheme : 1/16 Duty , ⅕ Bias
- Backlight (side) : LED (white)

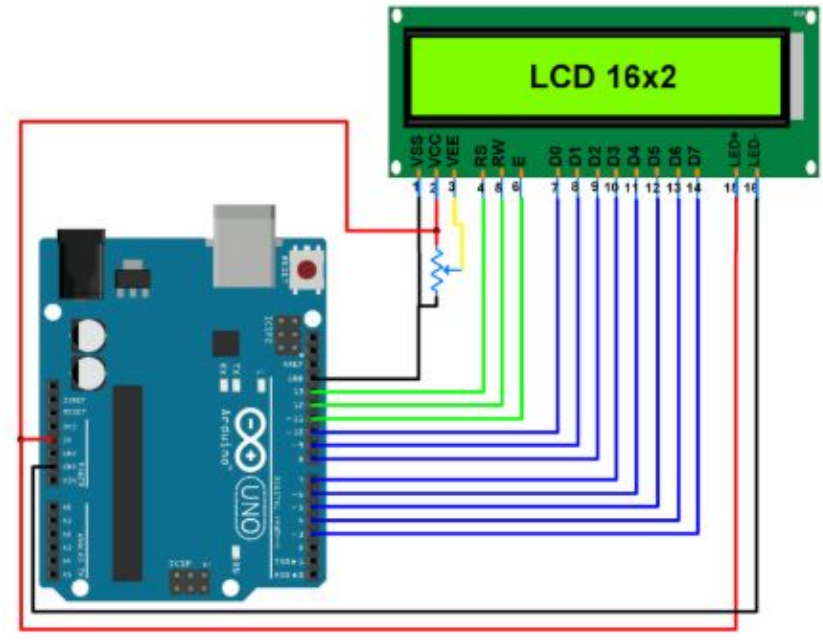
➢Pin Description :



No.	PIN	Function
1	VSS	Ground
2	VCC	+5 Volt
3	VEE	Contrast control 0 Volt: High contrast.

No.	PIN	Function
4	RS	Register Select 0: Commannd Reg. 1: Data Reg.
5	RW	Read / write 0: Write 1: Read
6	E	Enable H-L pulse
7-14	D0 - D7	Data Pins D7: Busy Flag Pin
15	LED+	+5 Volt
16	LED-	Ground

➢Interfacing with Arduino :



➢C Code :

```
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  Serial.begin(9600);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
  Serial.println(millis());
}
```

```
Done uploading.

Sketch uses 3414 bytes (1%) of program storage space. Maximum is 253952 bytes.
Global variables use 238 bytes (2%) of dynamic memory, leaving 7954 bytes for local variables. Maximum is 8192 bytes.
```

Size of code : 3414 bytes

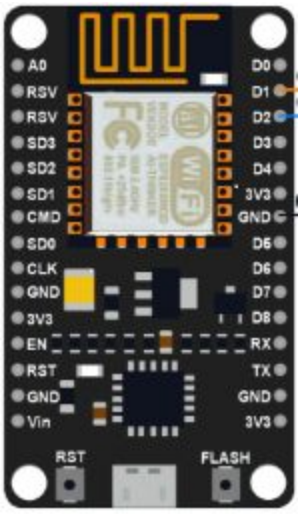
Execution Time : 25 ms

Chapter 9 Details of Supporting Tools

Details of additional modules :

1) ESP8266 - Wi-Fi Module :

NodeMCU is an open source LUA based firmware developed for ESP8266 wifi chip.

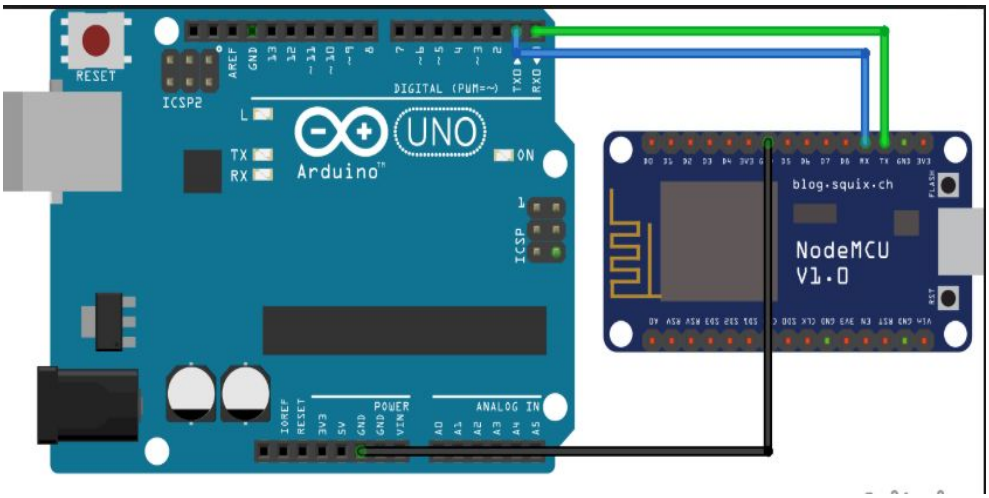


➢ Features :

- Programmable Wifi module
- Arduino-like(software defined) hardware IO.
- Can be programmed with the simple and powerful Lua programming language or Arduino IDE.
- USB-TTL included,plug and play.
- 10 GPIOs D0-D10, PWM functionality, IIC and SPI communication, 1-Wire and ADC A0 etc. all in one board.
- Wifi networking (can be used as access point and/or station, host a web server), concept to internet to fetch or upload data.

- Event-driven API for network applications.
- PCB antenna.

➢Interfacing with Arduino for Serial Communication :



➢Note :

We are using this Wifi module to store the records like in-time and out-time of the vehicle. Whenever the RFID reader will read the correct tag id, that vehicle's id and it's in-time and out-time will be written in the google drive spreadsheet. Thus, we can keep the track of records according to the time.

2) GSM (Global System for Mobile Communications) Module :

A GSM module or a GPRS module is a chip or circuit that will be used to establish communication between a mobile device or a computing machine and a GSM or GPRS system.

➢ **Features:**

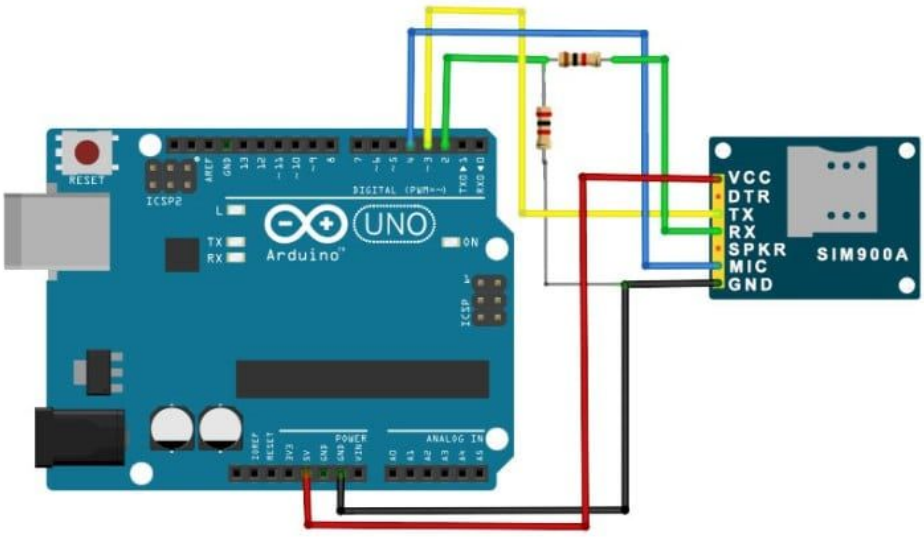
- Improved spectrum efficiency
- International roaming
- Compatibility with integrated services digital network (ISDN)
- Support for new services.
- SIM phonebook management
- Fixed dialing number (FDN)
- Real time clock with alarm management
- High-quality speech
- Uses encryption to make phone calls more secure

- Short message service (SMS)

GSM will allow communication anywhere, anytime, and with anyone. The functional architecture of GSM employing intelligent networking principles, and its ideology, which provides the development of GSM is the first step towards a true personal communication system that enough standardization to ensure compatibility.

GSM module can feature all the functionalities of a mobile phone through computer like making and receiving calls, SMS, MMS etc. These are mainly employed for computer based SMS and MMS services. These are mainly employed for computer-based SMS and MMS services.

➢Interfacing with Arduino for Serial Communication :



➢Note :

We are using this GSM module to send the message. Whenever the RFID reader will reads the wrong tag id, the owner of the warehouse will get the message that ‘Some unknown vehicle has tried to enter.’

Chapter 10 Complete Program

Complete C Program :

➢Arduino Mega Code

```
#include <Wire.h>
#include<LiquidCrystal.h>
// initialize the library by associating any needed LCD interface pin with the
arduino pin number it is connected to

#define colx 4

const int rn = 12, en = 11, d4 = 2, d5 = 3, d6 = 4, d7 = 5;   //Declare pins for lcd
LiquidCrystal lcd(rn, en, d4, d5, d6, d7);
int total = 0;

void setup()
{

  pinMode(22, INPUT);      //input pin for IRsensor1
  pinMode(24, INPUT);       //input pin for IRsensor2
  pinMode(26, INPUT);       //input pin for IRsensor3
  pinMode(28, INPUT);       //input pin for IRsensor4
  pinMode(30, INPUT);       //input pin for IRsensor5

  Serial.begin(9600);

// set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
```

```
  lcd.setCursor(0, 0);
  lcd.print("S1:");
  lcd.setCursor(6, 0);
  lcd.print("S2:");
  lcd.setCursor(12, 0);
  lcd.print("S3:");
  lcd.setCursor(0, 1);
  lcd.print("S4:");
  lcd.setCursor(6, 1);
  lcd.print("S5:");
}

void loop()
{

    int senread1 = digitalRead(22);   // Read the pin
    int senread2 = digitalRead(24);
    int senread3 = digitalRead(26);
    int senread4 = digitalRead(28);
    int senread5 = digitalRead(30);
    //int senread6 = digitalRead(32);

    lcd.setCursor(3,0) ;  // set up the location on LCD
    if(senread1 == !HIGH)   // if sensor1 detects something
    {
       lcd.print("N");     //  for indicating no space
       analogWrite(10,40); // LED will be low
    }
    else
    {
       lcd.print("Y");  // for the empty space
       analogWrite(10,255);   // LED will be High
    }
```

```
lcd.setCursor(9,0) ;  // set up the location on LCD
if(senread2 == !HIGH)
{
   lcd.print("N");
   analogWrite(9,40);
}
else
{
   lcd.print("Y");
   analogWrite(9,255);
}

lcd.setCursor(15,0) ; // set up the location in second row on LCD
if(senread3 == !HIGH)
{
   lcd.print("N");
   analogWrite(8,40);
}
else
{
   lcd.print("Y");
   analogWrite(8,255);
}

lcd.setCursor(3,1) ;  // set up the location in second row on LCD
if(senread4 == !HIGH)
{
   lcd.print("N");
   analogWrite(6,40);
}
else
{
```

```
        lcd.print("Y");
        analogWrite(6,255);
    }

    lcd.setCursor(9,1) ; // set up the location in second row on LCD
    if(senread5 == !HIGH)
    {
        lcd.print("N");
        analogWrite(7,40);
    }
    else
    {
        lcd.print("Y");
        analogWrite(7,255);
    }

}
```

➢Arduino Uno code : (For Entry)

```
//Arduino code
#include <Servo.h>
#include <SoftwareSerial.h>

#include "SPI.h"
#include "MFRC522.h"

#define SS_PIN 10
#define RST_PIN 9
//#define SP_PIN 52

Servo myservo;
```

```
MFRC522 rfid(SS_PIN, RST_PIN);

MFRC522::MIFARE_Key key;


SoftwareSerial s(5,6);
int pin = 7;
int pin1 = 3;
int sensor;
int data;

int pos = 0;


void setup() {
  pinMode(7,INPUT);
s.begin(9600);
Serial.begin(9600);
SPI.begin();
  rfid.PCD_Init();

  myservo.attach(2);
}

void loop() {

  /*for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
```

```
    delay(15);
  }
*/
if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial())
    return;

  // Serial.print(F("PICC type: "));
  MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
  // Serial.println(rfid.PICC_GetTypeName(piccType));

  // Check is the PICC of Classic MIFARE type
  if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
    piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
    piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println(F("Your tag is not of type MIFARE Classic."));
    return;
  }

  String strID = "";
  for (byte i = 0; i < 4; i++) {
    strID +=
    (rfid.uid.uidByte[i] < 0x10 ? "0" : "") +
    String(rfid.uid.uidByte[i], HEX) +
    (i!=3 ? ":" : "");
  }
  strID.toUpperCase();

  Serial.print("Tap card key: ");
  Serial.println(strID);



 // char id = strID.charAt();
```

```
 rfid.PICC_HaltA();
 rfid.PCD_StopCrypto1();

if(s.available()>0)
{
/* char character;
      character = strID;
      s.write(character);
      delay(10);*/
char ch;
if(strID == "C9:C5:8A:48" && digitalRead(7) == LOW)
{
 ch = 'A';
 s.write(ch);
 Serial.println(ch);
}
if(strID == "44:01:99:17" && digitalRead(7) == LOW)
{
 ch = 'B';
 for (pos = 0; pos <= 120; pos += 10) { // goes from 0 degrees to 180 degrees
   // in steps of 1 degree
   myservo.write(pos);
   delay(15);// tell servo to go to position in variable 'pos'
                    // waits 15ms for the servo to reach the position
 }
 s.write(ch);
 Serial.println(ch);
}
if(strID == "68:FD:7B:26" && digitalRead(7) == LOW)
{
 ch = 'C';
 s.write(ch);
 Serial.println(ch);
```

```
  for (pos = 0; pos <= 120; pos += 10) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
                    // waits 15ms for the servo to reach the position
  }

 }

if(digitalRead(3) == LOW)
 {
  for (pos = 0; pos <= 360; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
                     // waits 15ms for the servo to reach the position
  }
 }

}
delay(500);
}
```

➢NodeMCU code : (For Entry)

```
#include <ESP8266WiFi.h>
#include "HTTPSRedirect.h"
//#include <Wire.h>

#include <SoftwareSerial.h>
SoftwareSerial s(D6,D5);
char data;

//#include <Adafruit_MLX90614.h>
```

```
// Fill ssid and password with your network credentials
const char* ssid = "AU_STUDENT";
const char* password = "1111122222";

const char* host = "script.google.com";
const int httpsPort = 443;
const char *GScriptId =
"AKfycbxX3TL4l4zhHWrnpUWaX4PBjGXZ17jD0ZQawmmJW44T5eYeLfk";

// Write to Google Spreadsheet
//String url = String("/macros/s/") + GScriptId +
"/exec?temperature=adc_A0&humidity=";
String url = String("/macros/s/") + GScriptId + "/exec?id=";

String payload = "";



HTTPSRedirect* client = nullptr;
// used to store the values of free stack and heap
// before the HTTPSRedirect object is instantiated
// so that they can be written to Google sheets
// upon instantiation

//const int analog_ip = A0;
//int inputVal  = 0;
//Adafruit_MLX90614 mlx = Adafruit_MLX90614();

void setup() {

  s.begin(9600);
  Serial.begin(9600);
  Serial.flush();
```

```
Serial.println();
Serial.print("Connecting to wifi: ");
Serial.println(ssid);
//Wire.begin(D1,D2);
// flush() is needed to print the above (connecting...) message reliably,
// in case the wireless connection doesn't go through
Serial.flush();

WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

// Use HTTPSRedirect class to create a new TLS connection
client = new HTTPSRedirect(httpsPort);
client->setInsecure();
client->setPrintResponseBody(true);
client->setContentTypeHeader("application/json");

Serial.print("Connecting to ");
Serial.println(host);

// Try to connect for a maximum of 5 times
bool flag = false;
for (int i=0; i<5; i++){
  int retval = client->connect(host, httpsPort);
  if (retval == 1) {
```

```
      flag = true;
      break;
    }
    else
      Serial.println("Connection failed. Retrying...");
  }

  if (!flag){
    Serial.print("Could not connect to server: ");
    Serial.println(host);
    Serial.println("Exiting...");
    return;
  }

  payload = "temperature=aaaa&humidity=122";
  client->POST(url, host, payload, false);
  client->GET(url, host);
}

void loop() {
  static int connect_count = 0;
  static bool flag = false;

//int i = 1;
 /* Serial.print("Ambient = ");
  inputVal = mlx.readObjectTempC();
  Serial.print(inputVal);
  Serial.println("*C");*/
  s.write("s");
  if (s.available()>0)
  {
    data=s.read();
    Serial.println(data);
```

```
  String myString;
  if(data == 'A')
  {
   myString = String(data) + ":->Invalid";
  }
  else
  {
   myString = String(data);
  }

 //String myString2 = String(110);
 String FinalStringToSend;
 FinalStringToSend = url + myString;
 Serial.println(url);
 Serial.println(myString);
 //Serial.println(myString2);
 Serial.println(FinalStringToSend);


 if (client != nullptr){
  if (!client->connected()){
   client->connect(host, httpsPort);
    payload= "";
    Serial.println("POST Data to Sheet");
//     FinalStringToSend = url + myString;
 Serial.println("POST url :" + FinalStringToSend);

   client->POST(FinalStringToSend, host, payload);
  }
 }
 else{
 Serial.println(" >> Failed to POST data");
 }
```

```
  Serial.println("GET url :"+FinalStringToSend);
  client->GET(FinalStringToSend, host);
  Serial.println("Over");


  }
 delay(300);

}
```

➢Arduino Uno code : (For Exit)

```
//Arduino code
#include <Servo.h>
#include <SoftwareSerial.h>

#include "SPI.h"
#include "MFRC522.h"

#define SS_PIN 10
#define RST_PIN 9
//#define SP_PIN 52

Servo myservo;

MFRC522 rfid(SS_PIN, RST_PIN);

MFRC522::MIFARE_Key key;


SoftwareSerial s(3,4);
```

```
int pin = 7;
int pin1 = 3;
int sensor;
int data;

int pos = 0;


void setup() {
  pinMode(7,INPUT);
s.begin(9600);
Serial.begin(9600);
SPI.begin();
  rfid.PCD_Init();

  myservo.attach(2);
}

void loop() {

if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial())
    return;

  // Serial.print(F("PICC type: "));
  MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
  // Serial.println(rfid.PICC_GetTypeName(piccType));

  // Check is the PICC of Classic MIFARE type
  if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
    piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
    piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println(F("Your tag is not of type MIFARE Classic."));
    return;
```

```
  }

  String strID = "";
  for (byte i = 0; i < 4; i++) {
    strID +=
    (rfid.uid.uidByte[i] < 0x10 ? "0" : "") +
    String(rfid.uid.uidByte[i], HEX) +
    (i!=3 ? ":" : "");
  }
  strID.toUpperCase();

  Serial.print("Tap card key: ");
  Serial.println(strID);


 // char id = strID.charAt();

  rfid.PICC_HaltA();
  rfid.PCD_StopCrypto1();

    //sensor = digitalRead(pin);
    //Serial.println(sensor);
    //delay(10);
    //data=50;
    //Serial.println(data);
if(s.available()>0)
{
 char ch;
 if(strID == "C9:C5:8A:48" && digitalRead(7) == LOW)
 {
  ch = 'A';
  s.write(ch);
  Serial.println(ch);
```

```
 }
 if(strID == "44:01:99:17" && digitalRead(7) == LOW)
 {
  ch = 'B';
  s.write(ch);
  Serial.println(ch);
 }
 if(strID == "68:FD:7B:26" && digitalRead(7) == LOW)
 {
  ch = 'C';
  s.write(ch);
  Serial.println(ch);
 }

}
delay(500);
}
```

➢NodeMCU code : (For Exit)

```
#include <ESP8266WiFi.h>
#include "HTTPSRedirect.h"
//#include <Wire.h>

#include <SoftwareSerial.h>
SoftwareSerial s(D4,D3);
char data;

//#include <Adafruit_MLX90614.h>

// Fill ssid and password with your network credentials
const char* ssid = "AU_STUDENT";
```

```
const char* password = "1111122222";

const char* host = "script.google.com";
const int httpsPort = 443;
const char *GScriptId =
"AKfycbyfQUNMXkeXZDLnGOYzHtlDNPk_NoGsIYXP6IlAoexv5Jx1F00";

// Write to Google Spreadsheet
//String url = String("/macros/s/") + GScriptId +
"/exec?temperature=adc_A0&humidity=";
String url = String("/macros/s/") + GScriptId + "/exec?id=";

String payload = "";


HTTPSRedirect* client = nullptr;
// used to store the values of free stack and heap
// before the HTTPSRedirect object is instantiated
// so that they can be written to Google sheets
// upon instantiation

//const int analog_ip = A0;
//int inputVal  = 0;
//Adafruit_MLX90614 mlx = Adafruit_MLX90614();

void setup() {

  s.begin(9600);
  Serial.begin(9600);
  Serial.flush();

  Serial.println();
  Serial.print("Connecting to wifi: ");
```

```
Serial.println(ssid);
//Wire.begin(D1,D2);
// flush() is needed to print the above (connecting...) message reliably,
// in case the wireless connection doesn't go through
Serial.flush();

WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

// Use HTTPSRedirect class to create a new TLS connection
client = new HTTPSRedirect(httpsPort);
client->setInsecure();
client->setPrintResponseBody(true);
client->setContentTypeHeader("application/json");

Serial.print("Connecting to ");
Serial.println(host);

// Try to connect for a maximum of 5 times
bool flag = false;
for (int i=0; i<5; i++){
  int retval = client->connect(host, httpsPort);
  if (retval == 1) {
     flag = true;
     break;
  }
```

```
    else
      Serial.println("Connection failed. Retrying...");
  }

  if (!flag){
    Serial.print("Could not connect to server: ");
    Serial.println(host);
    Serial.println("Exiting...");
    return;
  }

  payload = "temperature=aaaa&humidity=122";
  client->POST(url, host, payload, false);
  client->GET(url, host);
}

void loop() {
  static int connect_count = 0;
  static bool flag = false;

//int i = 1;
 /* Serial.print("Ambient = ");
  inputVal = mlx.readObjectTempC();
  Serial.print(inputVal);
  Serial.println("*C");*/
  s.write("s");
  if (s.available()>0)
  {
    data=s.read();
    Serial.println(data);
    String myString;
    if(data == 'A')
    {
```

```
    myString = String(data) + ":->Invalid";
   }
   else
   {
    myString = String(data);
   }

  //String myString2 = String(110);
  String FinalStringToSend;
  FinalStringToSend = url + myString;
  Serial.println(url);
  Serial.println(myString);
  //Serial.println(myString2);
  Serial.println(FinalStringToSend);


  if (client != nullptr){
   if (!client->connected()){
    client->connect(host, httpsPort);
     payload= "";
    Serial.println("POST Data to Sheet");
//     FinalStringToSend = url + myString;
  Serial.println("POST url :" + FinalStringToSend);

    client->POST(FinalStringToSend, host, payload);
   }
  }
  else{
  Serial.println(" >> Failed to POST data");
  }
  Serial.println("GET url :"+FinalStringToSend);
  client->GET(FinalStringToSend, host);
  Serial.println("Over");
```

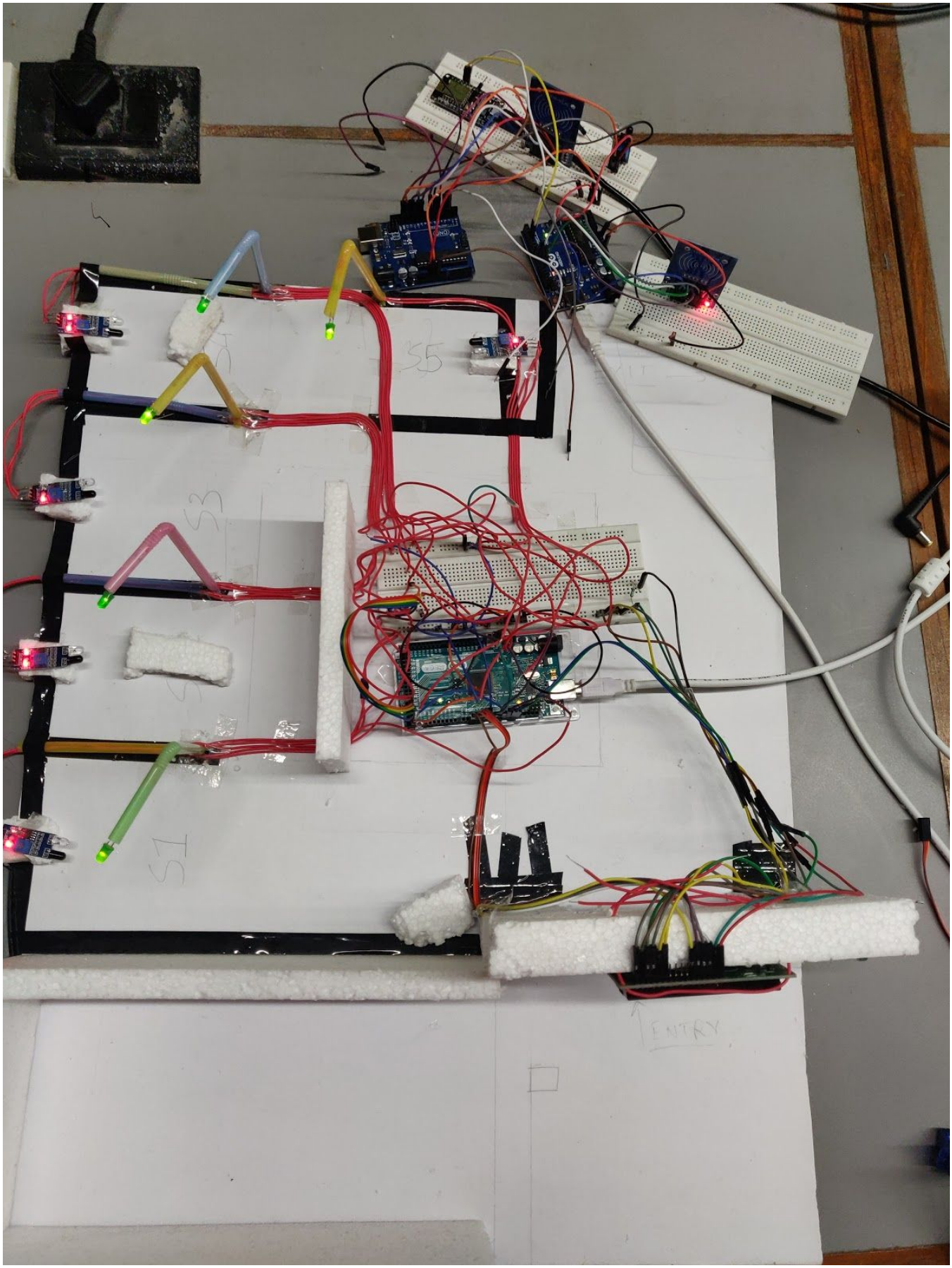
```
  }
 delay(300);

}
```

Chapter 11 Summary

Summary of the working of the project :

In our project we are providing the security system to the warehouse by the RFID tag and GSM module. RFID tag will detect the tag, and if it is correct then the servo motor will opens the gate. And according to the id, vehicle's id and it's in-time and out-time will store into the google drive spreadsheet. And also according to the IR sensors, the LCD display and the LED connected to the each parking slot will display, whether the parking slot is empty or not. By using this system, we are providing more efficient way of security. And if the RFID reader will detect the wrong tag id then the owner will get a message that 'Some unknown vehicle has tried to enter.' Thus by providing the datasheet, the owner of the warehouse can keep the track of the record of each vehicle.

As of now we are not able to detect the proper vehicle by its number plate. So, we can use this project in future by capturing the number plate of the vehicle at entrance and store it into the database to keep the track of records.


S5
S3
S1
ENTRY

Time-line:

	31/01/19	08/02/19	05/03/19	19/03/19	06/04/19
Deciding Topic	X				
Block Diagram	X	X			
Flow Chart		X			
Circuit Diagram			X		
Testing with NodeMCU and Arduino			X		
Testing with GSM Module and Arduino			X	X	
Code				X	
Combining And Testing				X	
Final Demonstration					X

Chapter 12 References

References :

[1] Serial Communication between NodeMCU and Arduino
Gowtham S March 19, 2018
https://mybtechprojects.tech/serial-communication-between-nodemcu-and-arduino/

[2] International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 17 (2017) pp. 6559-6563 © Research India Publications.
http://www.ripublication.com'https://www.ripublication.com/ijaer17/ijaerv12n17_35.pdf

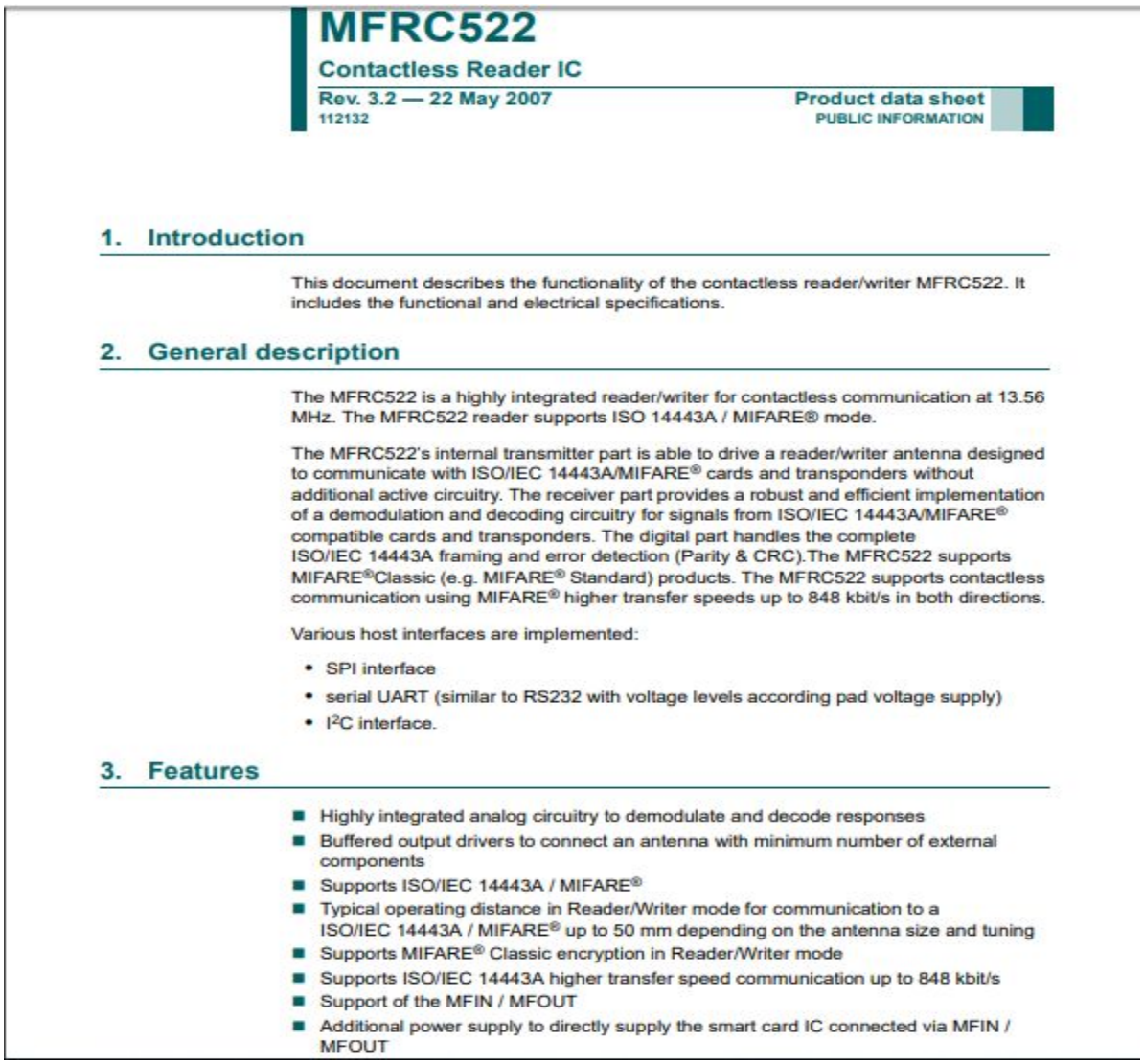
[3] How to send data from ESP8266 to Google Drive ? Published by Trieu Le on November 7,2017
http://lethanhtrieu.likesyou.org/2017/11/07/how-to-send-data-from-esp8266-to-google-drive/?i=1

[4] Arduino Tutorial: RFID Tutorial RC522 with an Arduino Uno and an OLED display from Banggood.com. Published on Oct 8, 2016
https://www.youtube.com/watch?v=So83sH6-jwM

Appendix A Datasheets

Selected Pages of Datasheets of all sensors and actuators used :

1) RFID tag and Reader :



MFRC522

Contactless Reader IC

Rev. 3.2 — 22 May 2007
112132

Product data sheet
PUBLIC INFORMATION

1. Introduction

This document describes the functionality of the contactless reader/writer MFRC522. It includes the functional and electrical specifications.

2. General description

The MFRC522 is a highly integrated reader/writer for contactless communication at 13.56 MHz. The MFRC522 reader supports ISO 14443A / MIFARE® mode.

The MFRC522's internal transmitter part is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443A/MIFARE® cards and transponders without additional active circuitry. The receiver part provides a robust and efficient implementation of a demodulation and decoding circuitry for signals from ISO/IEC 14443A/MIFARE® compatible cards and transponders. The digital part handles the complete ISO/IEC 14443A framing and error detection (Parity & CRC).The MFRC522 supports MIFARE®Classic (e.g. MIFARE® Standard) products. The MFRC522 supports contactless communication using MIFARE® higher transfer speeds up to 848 kbit/s in both directions.

Various host interfaces are implemented:

- SPI interface
- serial UART (similar to RS232 with voltage levels according pad voltage supply)
- I^2C interface.

3. Features

- Highly integrated analog circuitry to demodulate and decode responses
- Buffered output drivers to connect an antenna with minimum number of external components
- Supports ISO/IEC 14443A / MIFARE®
- Typical operating distance in Reader/Writer mode for communication to a ISO/IEC 14443A / MIFARE® up to 50 mm depending on the antenna size and tuning
- Supports MIFARE® Classic encryption in Reader/Writer mode
- Supports ISO/IEC 14443A higher transfer speed communication up to 848 kbit/s
- Support of the MFIN / MFOUT
- Additional power supply to directly supply the smart card IC connected via MFIN / MFOUT

2) IR Sensor :

1.Descriptions

The Multipurpose Infrared Sensor is an add-on for your line follower robot and obstacle avoiding robot that gives your robot the ability to detect lines or nearby objects. The sensor works by detecting reflected light coming from its own infrared LED. By measuring the amount of reflected infrared light, it can detect light or dark (lines) or even objects directly in front of it. An onboard RED LED is used to indicate the presence of an object or detect line. Sensing range is adjustable with inbuilt variable resistor.

The sensor has a 3-pin header which connects to the microcontroller board or Arduino board via female to female or female to male jumper wires. A mounting hole for easily connect one or more sensor to the front or back of your robot chassis.

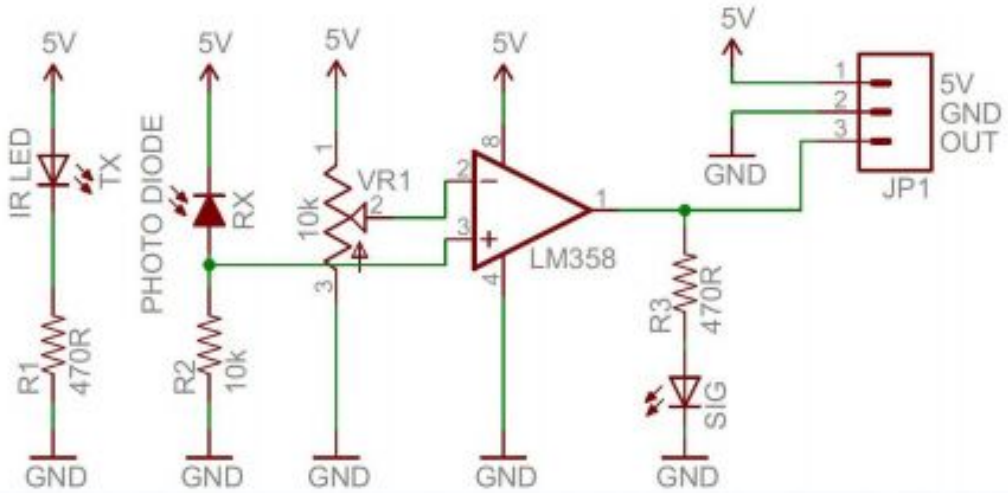
2.Features

- 5VDC operating voltage.
- I/O pins are 5V and 3.3V compliant.
- Range: Up to 20cm.
- Adjustable Sensing range.
- Built-in Ambient Light Sensor.
- 20mA supply current.
- Mounting hole.

3.Specifications

- Size: 50 x 20 x 10 mm (L x B x H)
- Hole size: ϕ2.5mm

4.Schematics



3) Servo Motor :

Servo Motor Data Sheet

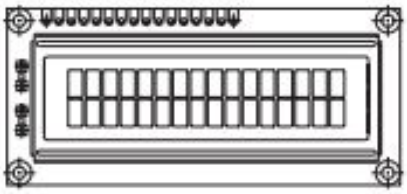
There are two types of Servo motors, continuous and sweep. More about SERVO Motors

Title	Servo Motor
TI Item Name	STEMKT/AC/D
Description	360 degree, continuous rotation servo motor with gearing and feedback system; used in driving mechanism of robots.
Category	Motors
Hub Connection	4-Pin Cable to only this port: OUT 3
Assembly Instructions	Mount a gear to the top of the Servo Motor using one of the provided screws.
Precautions	Use an Auxiliary Power Source. Do not hold the Servo Motor's shaft while it is rotating. Also, do not rotate the Servo Motor by hand.
Specifications	Operating Speed: 110RPM (4.8V), 130RPM (6V) Stall Torque: 1.3kg.cm/18.09oz.in (4.8V), 1.5kg.cm/20.86oz.in(6V) Operating Voltage: 4.8V~6V

HUB Commands	
Sketch Object	SERVO
Command Syntax	Send("SET SERVO n TO [CW/CCW] speed [[TIME] seconds] -- speed from -100 to 100, CW/CCW (Clockwise/Counterclosewise) optional, if speed <0, CCW, else CW unless CW/CCW keyword is specified, TIME optional, in seconds, default=1 second (for continuous servo operation) (CW/CCW required if TIME/seconds NOT specified.)

4) LCD Display :

16 x 2 Character LCD



FEATURES

- 5 x 8 dots with cursor
- Built-in controller (KS 0066 or Equivalent)
- + 5V power supply (Also available for + 3V)
- 1/16 duty cycle
- B/L to be driven by pin 1, pin 2 or pin 15, pin 16 or A.K (LED)
- N.V. optional for + 3V power supply

MECHANICAL DATA

ITEM	STANDARD VALUE	UNIT
Module Dimension	80.0 x 36.0	mm
Viewing Area	66.0 x 16.0	mm
Dot Size	0.56 x 0.66	mm
Character Size	2.96 x 5.56	mm

ABSOLUTE MAXIMUM RATING

ITEM	SYMBOL	STANDARD VALUE			UNIT
		MIN.	TYP.	MAX.	
Power Supply	VDD-VSS	- 0.3	–	7.0	V
Input Voltage	VI	- 0.3	–	VDD	V

NOTE: VSS = 0 Volt, VDD = 5.0 Volt

ELECTRICAL SPECIFICATIONS

ITEM	SYMBOL	CONDITION		STANDARD VALUE			UNIT
				MIN.	TYP.	MAX.	
Input Voltage	VDD	VDD = + 5V		4.7	5.0	5.3	V
		VDD = + 3V		2.7	3.0	5.3	V
Supply Current	IDD	VDD = 5V		–	1.2	3.0	mA
Recommended LC Driving Voltage for Normal Temp. Version Module	VDD - V0	- 20 °C		–	–	–	V
		0°C		4.2	4.8	5.1	
		25°C		3.8	4.2	4.6	
		50°C		3.6	4.0	4.4	
		70°C		–	–	–	
LED Forward Voltage	VF	25°C		–	4.2	4.6	V
LED Forward Current	IF	25°C	Array	–	130	260	mA
			Edge	–	20	40	
EL Power Supply Current	IEL	Vel = 110VAC:400Hz		–	–	5.0	mA

DISPLAY CHARACTER ADDRESS CODE:

Display Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DD RAM Address	00	01														0F
DD RAM Address	40	41														4F

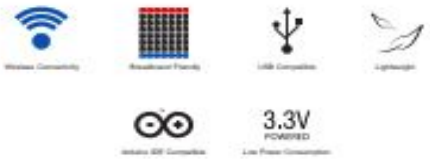
5) Wi-fi Module :

WiFi Development Board

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the DevKit. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson, and spiffs.

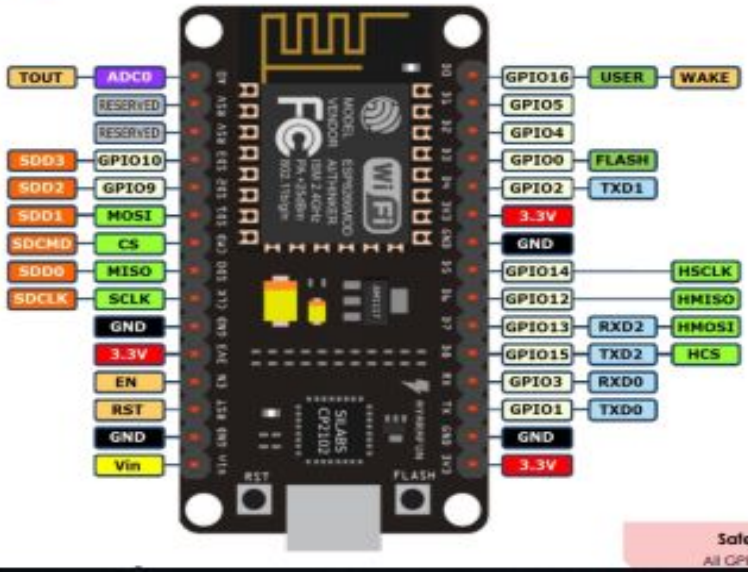
Features

- Version : DevKit v1.0
- Breadboard Friendly
- Light Weight and small size.
- 3.3V operated, can be USB powered.
- Uses wireless protocol 802.11b/g/n.
- Built-in wireless connectivity capabilities.
- Built-in PCB antenna on the ESP-12E chip.
- Capable of PWM, I2C, SPI, UART, 1-wire, 1 analog pin.
- Uses CP2102 USB Serial Communication interface module.
- Arduino IDE compatible (extension board manager required).
- Supports Lua (alike node.js) and Arduino C programming language.



PINOUT DIAGRAM

NodeMCU ESP8266 v1.0



Safety Precaution

All GPIO runs at 3.3V !!

6) GSM Module :

GSM Shield

1 Description:

The GSM shield by Arduino is used to send/ receive messages and make/receive calls just like a mobile phone by using a SIM card by a network provider. We can do this by plugging the GSM shield into the Arduino board and then plugging in a SIM card from an operator that offers GPRS coverage.

The shield employs the use of a radio modem by SIMComm. We can communicate easily with the shield using the **AT** commands. The **GSM library** contains many methods of communication with the shield.

This GSM Modem can work with any GSM network operator SIM card just like a mobile phone with its own unique phone number. Advantage of using this modem will be that its RS232 port can be used to communicate and develop embedded applications. Applications like SMS Control, data transfer, remote control and logging can be developed easily using this.

The modem can either be connected to PC serial port directly or to any microcontroller through MAX232. It can be used to send/receive SMS and make/receive voice calls. It can also be used in GPRS mode to connect to internet and run many applications for data logging and control. In GPRS mode you can also connect to any remote FTP server and upload files for data logging

This GSM modem is a highly flexible plug and play quad band **SIM900A** GSM modem for direct and easy integration to RS232 applications. It Supports features like **Voice, SMS, Data/Fax, GPRS and integrated TCP/IP stack.**

To be connected to a cellular network, the shield requires a SIM card provided by a network provider.

Most recent revision of the board makes the connection of the shield with the Arduino Uno board by connecting its **TX** to **pin 0** of Arduino and **pin 1** of Arduino to **RX** of shield.

1.2 Features:

- Dual-Band 900/ 1800 MHz
- GPRS multi-slot class 10/8
- GPRS mobile station class B
- Compliant to GSM phase 2/2+
  - Class 4 (2 W @900 MHz)
  - Class 1 (1 W @ 1800MHz)
- Dimensions: 24*24*3 mm
- Weight: 3.4g
- Control via AT commands (GSM 07.07, 07.05 and SIMCOM enhanced AT Commands)
- SIM application toolkit
- Supply voltage range: 3.1- 4.8V
- Low power consumption: 1.5mA(sleep mode)
- Operation temperature: -40° C to +85°C

1.3 Specifications for Fax:

- Group 3, class 1
- Specifications for Data
- GPRS class 10: Max. 85.6 kbps (downlink)
- PBCCH support
- Coding schemes CS 1, 2, 3, 4
- CSD up to 14.4 kbps
- USSD
- Non transparent mode
- PPP-stack

1.4 Specifications for SMS via GSM/GPRS:

- Point to point MO and MT
- SMS cell broadcast
- Text and PDU mode

Appendix B Programming Review

Compare C Programming with two other language :

Comparison between C and Python :

C	Python
C is mainly use for hardware related application.	Python is general purpose programming language.
Follows an imperative programming model.	Follows object oriented programming language.
C is compiled.	Python is interpreted.
A limited number of built in functions.	Large library of built in functions.
Code execution is faster than python.	Slower compared to C,as python has garbage collection.
Implementing data structures required its functions to explicitly implemented.	Gives ease of implementing data structures with built in insert,append functions.
It is compulsory to declare the variable type in C.	No need to declare a type of variable.
C program syntax is harder than python.	Python programs are easier to learn,write and read.

Comparison between C and Assembly language :

C	Assembly language
C code can be easily reused on a different platform.	Assembly does not provide the portability and source code specific to a processor.
Execution speed and manufacturing cost is more.	Execution speed and manufacturing cost is comparatively very less.
Requires lesser memory.	Requires larger memory.
It is easily portable to the other microcontroller with almost no modifications.	Not easily portable with the other microcontrollers.

Appendix C Trouble-shooting

Trouble-shooting and debugging the project work :

- As we want to send data to google spreadsheet using Wi-fi module, read by the arduino we have to do serial communication for that. But, we are not able to **communicate between Arduino Mega and ESP8266 Wi-fi module.** So, for that we have first tried a simple code to te send int data from arduino to Wifi module using TX and RX pin. Then we have tried a code to read the sensor value from the arduino and send it to the Wi-fi module. But, because of the time delay we are able to get the data on the Wi-fi module but not in the correct time. So after doing some trial and error method we are able to manage time delay. But after doing that we have tried this using RFID. But also in that we are not able to pass the string in serial communication. So for that we have stored the string in one unique character and pass it. So for doing that we are able to optimize the code by comparing the character instead of string for the further process.

- We have to use 2 RFID reader, 2 IR sensor , 1 Wi-fi module ESP8266 and 1 servo motor in one Arduino uno. But because of the many components **Arduino takes too much load** , servo motor is not working properly. So for that we are giving external voltage and we are using one another Arduino Uno for connecting the other RFID reader.

- For sending data (RFID Tag, In-Time, Out-Time) to google spreadsheet, we were getting **error for connecting to host**. But after the proper connection the internet

we are able to send those data. And we have used characters to represent the tag-id instead of string.

- When we create the whole connections, we are not able to connect the GSM module !