ARDUINO CONTROLLER PROGRAMMING & ITS APPLICATIONS

Day-Five

Zuhaib A. Shaikh,
Asst. Prof., CSE Deptt.,QUEST
Web: zuhaib-shaikh.neocities.org

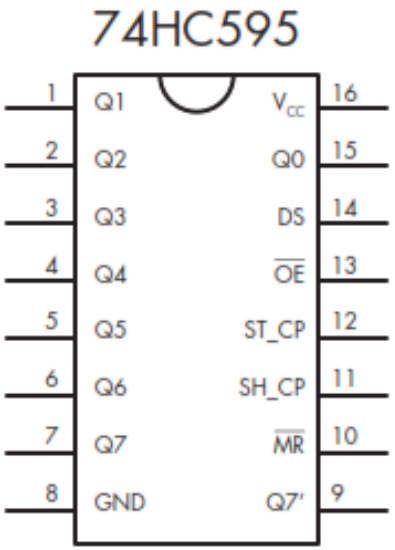
Outline

Day	Activity
1	i. Introduction to Arduino development board and Arduino IDE ii. C/C++ language overview iii. Basic input / output with Arduino iv. Overview of Proteus simulation software for Arduino simulation
2	i. Interfacing and glowing LEDs with different pattern ii. Interfacing push button and piezo buzzer iii. Interfacing a temperature sensor with Arduino
3	i. Familiarization with Serial Monitor for input and for output ii. Interfacing LDR sensor with Arduino iii. Interfacing PIR motion sensor with Arduino
4	i. Interfacing Arduino with LCD (16x2), relay and Servo motor ii. Interface Arduino with Sonic Sensor for obstacle detection
5	i. Interfacing shift register and 7-segment display with Arduino ii. Interfacing HC-05 Bluetooth module with Arduino iii. Driving GSM modem with Arduino

Prerequisites:
- Knowledge of C++
- Knowledge of basic electronic components

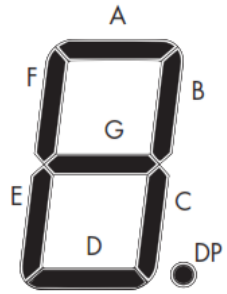
Shift Register 74HC595

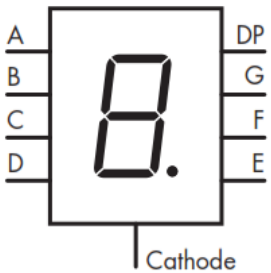
- Arduino has limited number of digital pins
- Digital pins can be extended using a Shift Register e.g. 74HC595
  - Analog & digital pins can also be extended using multiplexer i.e. 74HC4051
- 74HC595 pin configuration
  - Pins 15 and 1 to 7 are the eight output pins that we control (labeled Q0 to Q7, respectively).
    - Q7 outputs the first bit sent to the shift register, down to Q0, which out- puts the last
    - E.g. The data 10001001 will set Q7, Q4 and Q0 are high while other pins will be set to low
  - Pin 8 connects to GND
  - Pin 9 is “data out” and is used to send data to another shift register if one is present
  - Pin 10 is always connected to 5 V (for example, the 5 V connector or
  - Pins 11 and 12 are called clock and latch
  - Pin 13 is called output enable and is usually connected to GND
  - Pin 14 is for incoming bit data sent from the Arduino
  - Pin 16 is used for power: 5 V from the Arduino



7-segment Display

- Combination of 8 LEDs
- Usually used in many digital components
  - Digital Clock, digital meter etc.
  - Two major types
    - Common Cathode
    - Common Anode
- Common Cathode 7-segment Display
  - Pin A to G are used to display particular segment
  - DP (decimal point) is used to display decimal value
  - Cathode pin will be connected to GND
- The table shows the combination of the pins to generate desired output

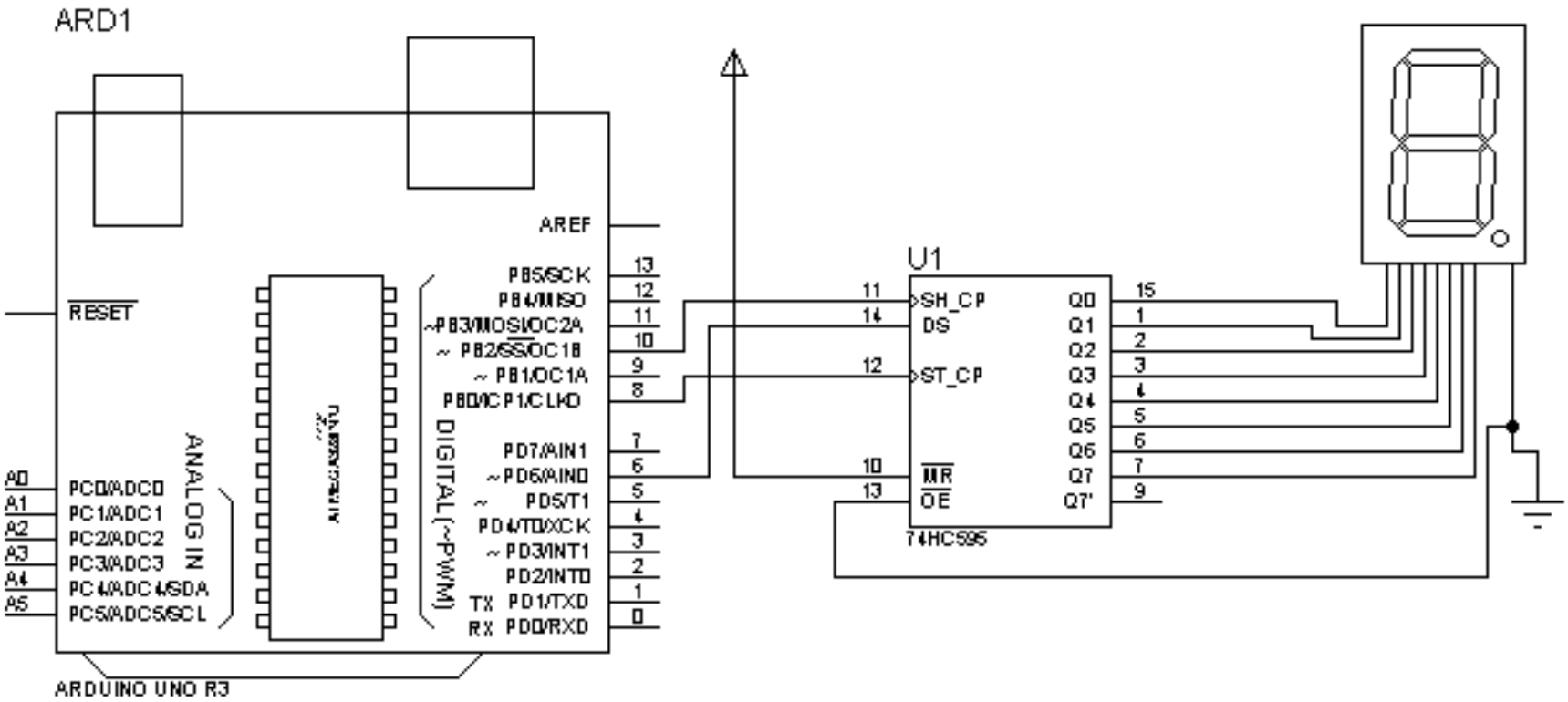




SR	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	
Segment	A	B	C	D	E	F	G	DP	Decimal
0	1	1	1	1	1	1	0	0	252
1	0	1	1	0	0	0	0	0	96
2	1	1	0	1	1	0	1	0	218
3	1	1	1	1	0	0	1	0	242
4	0	1	1	0	0	1	1	0	102
5	1	0	1	1	0	1	1	0	182
6	1	0	1	1	1	1	1	0	190
7	1	1	1	0	0	0	0	0	224
8	1	1	1	1	1	1	1	0	254
9	1	1	1	1	0	1	1	0	246
A	1	1	1	0	1	1	1	0	238
B	0	0	1	1	1	1	1	0	62
C	1	0	0	1	1	1	0	0	156
D	0	1	1	1	1	0	1	0	122
E	1	0	0	1	1	1	1	0	158
F	1	0	0	0	1	1	1	0	142

Example 1: 7-segment Display with Shift Register

Circuit



Example 1: 7-segment Display with Shift Register

Code

```
#define DATA  6                    // connect to pin 14 on the 74HC595
#define LATCH 8                    // connect to pin 12 on the 74HC595
#define CLOCK 10                   // connect to pin 11 on the 74HC595
int digits[] = {252, 96, 218, 242, 102, 182, 190, 224, 254, 246, 238, 62, 156, 122, 158, 142};
void setup()
{
  pinMode(LATCH, OUTPUT);
  pinMode(CLOCK, OUTPUT);
  pinMode(DATA, OUTPUT);
}
void loop()
{
    for (int i = 0 ; i < 16 ; i++ ) {
    digitalWrite(LATCH, LOW);
    shiftOut(DATA, CLOCK, LSBFIRST, digits[i]);
    digitalWrite(LATCH, HIGH);
    delay(250);
    }
  for (int i = 0 ; i < 16 ; i++ )  {
    digitalWrite(LATCH, LOW);
    shiftOut(DATA, CLOCK, LSBFIRST, digits[i]+1);
    digitalWrite(LATCH, HIGH);
    delay(250);
    }
}
```

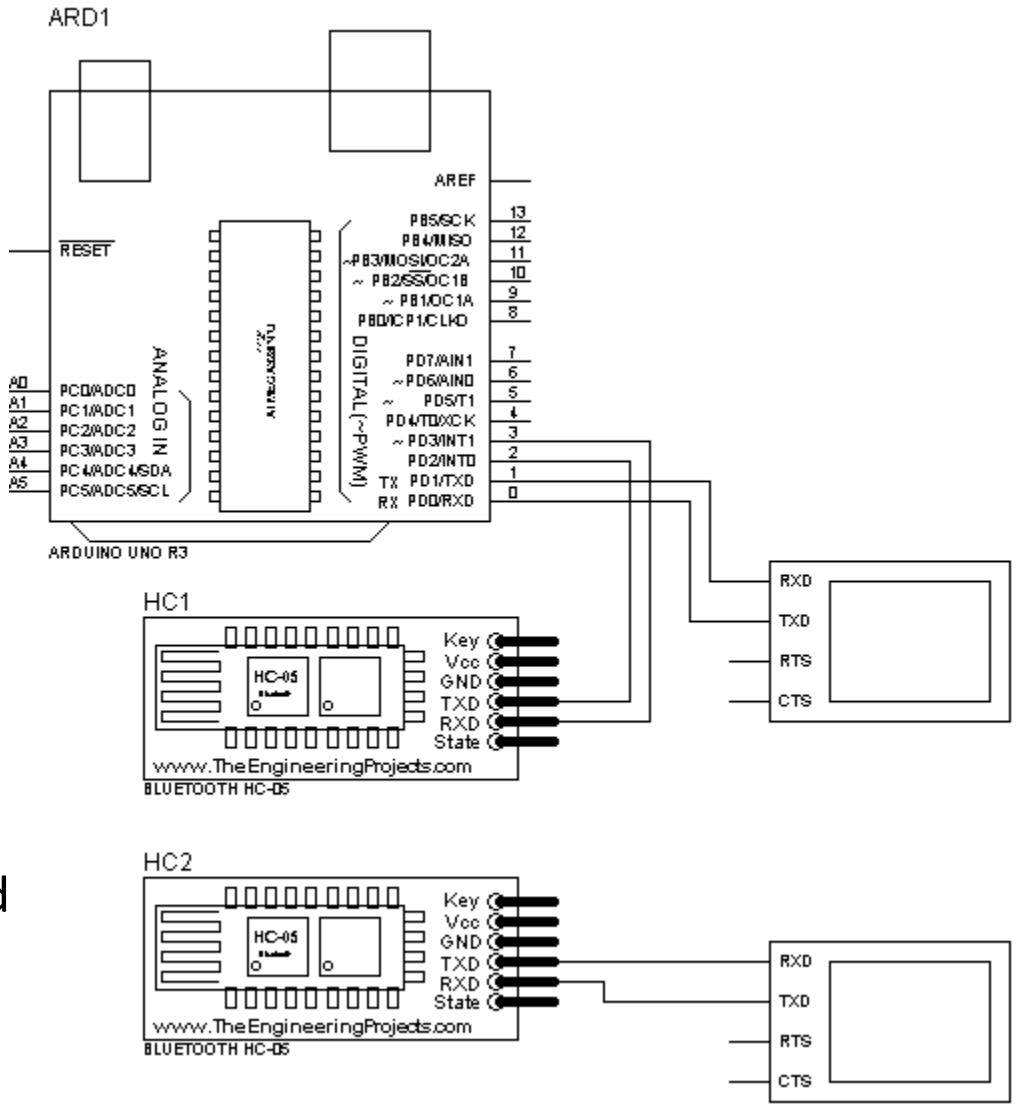
Exercise

- Simulate two 7-segment display with two shift registers using Arduino
  - Connect Pin 9 (Q7') of 1st shift register to Pin 14 (DS) of 2nd shift register
  - Two shift register can receive 2 bytes of data, 1st byte (L) for 1st register and 2nd byte (H) for 2nd register
  - e.g.

```
digitalWrite(LATCH, LOW);
shiftOut(DATA, CLOCK, MSBFIRST, 254); // data for first 74HC595
shiftOut(DATA, CLOCK, MSBFIRST, 254); // data for second 74HC595
digitalWrite(LATCH, HIGH);
```

Example 2: Bluetooth Module HC-05 with Arduino

- HC-05 Bluetooth Module
- It can be connected as:
  - Master
  - Slave
- Baud rate: 38400
- Pin configuration:
  - TX – Transmitting pin
  - RX – Receiving pin
  - VCC – +5v
  - GND – Ground
  - Key – To write AT commands for configuration
  - State – State of HC05
- To pair those virtually, Virtual port software will be required

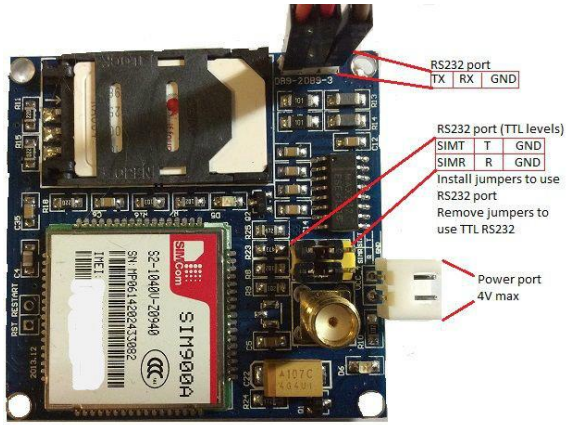


Example 2: Bluetooth Module HC-05 with Arduino

Code

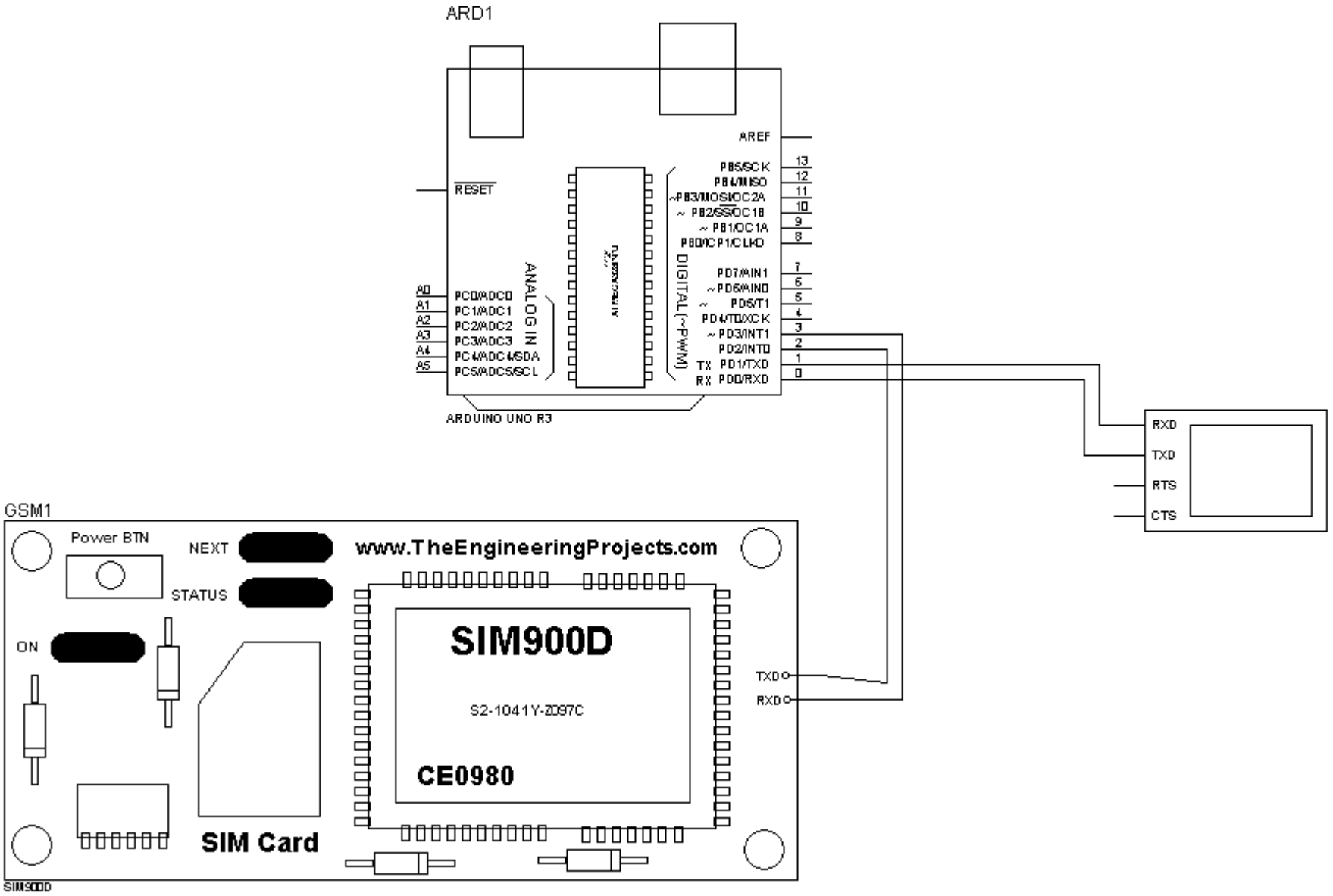
```
#include<SoftwareSerial.h>
SoftwareSerial BT(2,3);
void setup() {
Serial.begin(9600);
BT.begin(9600);
}

void loop() {
  if(Serial.available())
    BT.write(Serial.read());
  if(BT.available())
    Serial.write(BT.read());
}
```

GSM SIM900D Module



- SIM900A mini a popular GSM/GPRS module
- Indicator LED:
  - Blinks once per second – Trying to connect with network
  - Blinks once per 3 seconds – Connected
  - Power LED and indicator LED blinks simultaneously then high power is provided (than 4V, 0.2A)
  - Power LED and indicator LED resets means low power is provided
- Works on AT commands
- AT commands
  - AT – Checking module
  - AT+CMGF=1 – Setting to SMS mode
  - AT+CMGS=+923003218049\r – Setting phone number with \r end
    - String message to be written ending with Ctrl+z
  - AT+CNMI=2,2,0,0,0 – Read incoming message
  - ATD+923003218049; – Making Call
  - ATH – Call hang-up
  - ATA – Receive incoming call
  - ATDL – Redial call

Example 3: SIM900D GSM Module



Example 3: GSM SIM900D

Code

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);
char msg;
char call;
void setup()
{
  mySerial.begin(9600);
  Serial.begin(9600);
  Serial.println("SIM900A Operations");
  Serial.println("Enter character for control option:");
  Serial.println("h : to disconnect a call");
  Serial.println("i : to receive a call");
  Serial.println("s : to send message");
  Serial.println("c : to make a call");
  Serial.println("e : to redial");
  Serial.println();
  delay(100);
}
```

```
void loop()
{
  if (Serial.available()>0)
   switch(Serial.read())
  {
    case 's':
      SendMessage();
      break;
    case 'c':
      MakeCall();
      break;
    case 'h':
      HangupCall();
      break;
    case 'e':
      RedialCall();
      break;
    case 'i':
      ReceiveCall();
      break;
  }
 if (mySerial.available()>0)
 Serial.write(mySerial.read());
}
```

Example 3: GSM SIM900D

Code

```
void SendMessage()
{
  mySerial.println("AT+CMGF=1");
  delay(1000);
mySerial.println("AT+CMGS=\"+923003218049\"\r");
delay(1000);
  mySerial.println("Message from Arduino");
  delay(100);
   mySerial.println((char)26);
  delay(1000);
}
void ReceiveMessage()
{
  mySerial.println("AT+CNMI=2,2,0,0,0");
  delay(1000);
  if (mySerial.available()>0)
  {
    msg=mySerial.read();
    Serial.print(msg);
  }
}
void MakeCall()
{
  mySerial.println("ATD+923003218049;");
Serial.println("Calling  ");
  delay(1000);
}
void HangupCall()
{
  mySerial.println("ATH");
  Serial.println("Hangup Call");
  delay(1000);
}
void ReceiveCall()
{
  mySerial.println("ATA");
  delay(1000);
  {
    call=mySerial.read();
    Serial.print(call);
  }
}
void RedialCall()
{
  mySerial.println("ATDL");
  Serial.println("Redialing");
  delay(1000);
}
```

Questions