```
#include <SoftwareSerial.h> /* Library of Using Software Serial */
#include <String.h>  /* String Library */
#include <SPI.h>
#include <MFRC522.h>

/* Enable the use of Timer 2 inteerupt */
#define USE_TIMER_1     false
#define USE_TIMER_2     true
#define USE_TIMER_3     false
#define USE_TIMER_4     false
#define USE_TIMER_5     false

#include "TimerInterrupt.h" /* Timer interrupt library */

#define TIMER1_INTERVAL_MS    100 /* Compare match timer value */

SoftwareSerial SIM900(7,8); // Configura el puerto serial para el SIM GSM

//########################_FLOW_VARIABLES_########################//
#define FLOW_TICKS 1
#define FLOW_CALIBRATION (float)4.3
volatile int flow_measure;  /* Flow measaure interrupt counter */
int flow_measure_pin = 2; //Pin ubicación del sensor
float flow_measure_sum = 0; /* Sum of all the flow measure in a single minute */
float flow_measure_total = 0; /* The value measure during the last minute */

const int _PROMEDIO = 60; /* 60 Seconds */

int doorstate = 1; /*variable de puerta*/

//########################_RFID_AND_SOLENIDE#######################//
```

```
#define RST_PIN         9          // Configurable, see typical pin layout above
#define SS_PIN          10         // Configurable, see typical pin layout above
#define _SOLENOIDE 6
MFRC522 rfid(SS_PIN, RST_PIN); // Instance of the class
// Init array that will store new NUID
byte OK[4]={0x57,0x83,0x0B,0xD9};

//########################_ELECTRO_VALVULA_######################//
#define _VALVE 4 /* Pin of the valve */
String caracter="";// Variable para guardar los caracteres de mensajes entrantes, para accionar la valvula.

//########################_BUTTON_VARIABLES_######################//
#define _BUTTON 3 /* Pin of the button */
int variable_button = 1; /* Button state */
int button;

//########################_SIM900_VARIABLES_######################//
char incoming_char=0; //Variable que guarda los caracteres que envia el SIM GSM
int salir = 0;
unsigned long msg = 0;

//#########################_INTERRUPTS_#########################//
unsigned long timer1 = 0;          /* Timer interrupt counter */
unsigned long readflowcounter = 0; /* Number of interrupts since last time read flow operation was finished */
unsigned long senddatacounter = 0; /* Number of interrupts since last time send data operation was finished */
unsigned long loopcounter = 0;     /* Number of interrupts since last time the loop finished reading and sending data while the button is not pressed */
unsigned char loopstate = 0;       /* state variable of the loop function state machine */
void flow_sensor_int ()     //Interrupción para interupcion del sensor
```

```
{
  flow_measure++;          //Medidor de la rapidez del sensor
}



void button_int()
{
  button = digitalRead(_BUTTON);
  if(button == HIGH)
  {
    digitalWrite(_VALVE,LOW);
    variable_button = 0;
    Serial.print("INTRUSO DETECTADO\r\n");
  }
}
/* Timer ISR is triggered every 100ms */
void TimerHandler1(void)
{
  timer1++;
}



//############################_FUNCTIONS_############################//

float OPEN_OR_CLOSE_DOOR (){



  // Reset the loop if no new card present on the sensor/reader. This saves the entire process when idle.
  if ( ! rfid.PICC_IsNewCardPresent())
    return;
```

```
 // Verify if the NUID has been readed
 if ( ! rfid.PICC_ReadCardSerial())
   return;

 if (rfid.uid.uidByte[0] == OK[0] ||
   rfid.uid.uidByte[1] == OK[1] ||
   rfid.uid.uidByte[2] == OK[2] ||
   rfid.uid.uidByte[3] == OK[3] ) {

  if(doorstate==0){
   Serial.println(("CerrarPuerta"));
   analogWrite(_SOLENOIDE, 255);
   delay(80);
   analogWrite(_SOLENOIDE, 180);
  }
  if(doorstate==1){
   Serial.println(("AbrirPuerta"));
   analogWrite(_SOLENOIDE,55);
   delay(10);
   analogWrite(_SOLENOIDE,35);
   delay(10);
   analogWrite(_SOLENOIDE,15);
   delay(10);
   analogWrite(_SOLENOIDE,0);
  }

  doorstate=!doorstate;
 }

}
```

```
float read_flow_sensor () {
  float temp_total = 0;
  temp_total = (flow_measure / FLOW_CALIBRATION); //Formula de litro por minutos L/MIN
  temp_total = temp_total/FLOW_TICKS;
  flow_measure = 0;
  return temp_total;
}

/* Read data sent from SIM900 module to the Arduino */
void ShowSerialData(){
  while(SIM900.available()!=0)
  Serial.write(SIM900.read());
}

/* Send the measured data to Thingspeak server using SIM900 */
void send_data(){
  /* Get the time since last finished sending */
  unsigned long tempcounter = timer1 - senddatacounter;
  /* State variable of the state machine */
  static unsigned char state = 0;
  /* Read the buffer if there something in the software UART still not read */
    if (SIM900.available())
    Serial.write(SIM900.read());
  /* STATE 0: Starting state. We initate connection the server
   * STATE 1: Sending state. We initate the send request
   * STATE 2: Requesting state. We submit the GET request with the data
   * STATE 3: Finishing state. We conclude the request
   * STATE 4: Shutting staate. We shut down the connection
   * STATE 5: End state: We wait for the connection to be shut and reset the variables
   */
```

```
 if(tempcounter < 610 && state == 0)
 {
  SIM900.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"");//Iniciar conexión
  state = 1;
 }

 else if(tempcounter > 660 && tempcounter < 670 && state == 1)
 {

  ShowSerialData();

  SIM900.println("AT+CIPSEND");//Enviar datos al servidor
  state = 2;
 }
 else if(tempcounter > 720 && tempcounter < 730 && state == 2)
 {
  ShowSerialData();
  String str;
  if(String(flow_measure_total).length() > 4)
    str="GET https://api.thingspeak.com/update?api_key=O66W3QPOABYJIE8D&field1=" +
String(flow_measure_total).substring(0,4)+"&field2="
+String(flow_measure_total).substring(0,4)+"&field3="
+String(flow_measure_total).substring(0,4);
  else
    str="GET https://api.thingspeak.com/update?api_key=O66W3QPOABYJIE8D&field1=" +
String(flow_measure_total).substring(0,4)+"&field2="
+String(flow_measure_total).substring(0,4)+"&field3="
+String(flow_measure_total).substring(0,4);

  Serial.println(str);
  SIM900.println(str);
  state = 3;
 }
```

```
  else if(tempcounter > 780 && tempcounter < 790 && state == 3)
  {
    ShowSerialData();

    SIM900.println((char)26);//Enviando
    state = 4;
  }
  else if(tempcounter > 840 && tempcounter < 850 && state == 4)
  {
    SIM900.println();

    ShowSerialData();

    SIM900.println("AT+CIPSHUT");//Cerrar conexión
    state = 5;
  }
  else if(tempcounter > 900 && tempcounter < 910 && state == 5)
  {
    ShowSerialData();
    state = 6;
  }
  if(tempcounter > 900 && state == 6)
  {
    senddatacounter = timer1;
    loopcounter = timer1;
    state = 0;
    loopstate = 0;
    flow_measure_sum = 0;
  }
}
```

```
void AbrirCerrar_VALVE()
{

  if(SIM900.available()>0)
  {

    //Serial.println("Texto recibido SMS");

    //Verificamos si hay datos disponibles desde el SIM900
    caracter=SIM900.readString(); // Leemos los datos
    Serial.print(caracter); //Imprime datos entrantes

  if(caracter.indexOf("Cerrar")>0)
  {
    Serial.println("Cerrando Valvula");
    digitalWrite(_VALVE,LOW);
    delay(100);
    //caracter = "";
  }

  if(caracter.indexOf("Abrir")>0)
  {
    Serial.println("Abriendo Valvula");
    digitalWrite(_VALVE, HIGH);
    delay(100);
    //caracter = "";
  }
  }
}
```

```
//##############################_MAIN_############################//
void setup() {


  SIM900.begin(19200); //Configura velocidad serial para el SIM
  Serial.begin(19200);

  Serial.print(" INICIO PROTOTIPO_V1.0\r\n");



  while (!Serial) {
    ; // wait for serial port to connect. Needed for Native USB only
  }
  SIM900.print("AT+CMGF=1\r");// comando AT para configurar el SIM900 en modo texto
  delay(200);
  SIM900.print("AT+CNMI=2,2,0,0,0\r"); //Configuramos el módulo para que muestre los SMS por el puerto serie.
  delay(200);

  pinMode(_VALVE,OUTPUT);
  digitalWrite(_VALVE,HIGH);

  pinMode(flow_measure_pin,INPUT);
  pinMode(_BUTTON,INPUT);

  analogWrite(_SOLENOIDE, 190);
  delay(80);
  analogWrite(_SOLENOIDE, 80);

  MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance RFID
  SPI.begin();      // Init SPI bus
```

```
mfrc522.PCD_Init();   // Init MFRC522

rfid.PCD_Init(); // Init MFRC522
pinMode(_SOLENOIDE,OUTPUT);

attachInterrupt(0, flow_sensor_int, RISING);
//attachInterrupt(digitalPinToInterrupt(3), button_int, FALLING);

SIM900.println("AT");
delay(1000);

SIM900.println("AT+CPIN?");
delay(1000);

SIM900.println("AT+CREG?");
delay(1000);

SIM900.println("AT+CGATT?");
delay(1000);

SIM900.println("AT+CIPSHUT");
delay(1000);

SIM900.println("AT+CIPSTATUS");
delay(2000);

SIM900.println("AT+CIPMUX=0");
delay(2000);

ShowSerialData();
```

```
SIM900.println("AT+CSTT=\"claro\"");//Iniciar APN
delay(1000);

ShowSerialData();

SIM900.println("AT+CIICR");//abrir conexión inalambrica
delay(3000);

ShowSerialData();

SIM900.println("AT+CIFSR");//Obtener dirección IP
delay(2000);

ShowSerialData();

SIM900.println("AT+CIPSPRT=0");
delay(3000);

ShowSerialData();

sei();

ITimer2.init();
if (ITimer2.attachInterruptInterval(TIMER1_INTERVAL_MS, TimerHandler1))
{
  Serial.print(F("Starting  ITimer1 OK, millis() = ")); Serial.println(millis());
}
else
  Serial.println(F("Can't set ITimer1. Select another freq. or timer"));

timer1 = 0;
```

```
}

void loop() {

 button_int();
 AbrirCerrar_VALVE();



 /* Calculate the number of interrupts since the last time the flow measurement was read */
 unsigned long tempcounter = timer1 - readflowcounter;
 /* Calculate the number of interrupts since the last time the data sending was initiated */
 unsigned long tempcounter2 = timer1 - loopcounter;
 float flow_measure_local = 0;
 switch(variable_button) {
  case 1:
  /* Caclulate the reading every 1 second
   * And send the data every 1 minute
   */
  /* If one second has passed calculate the flow during the last second */
  if(tempcounter > 10)
  {
    flow_measure_local = read_flow_sensor();
    flow_measure_sum = flow_measure_sum + flow_measure_local;
    readflowcounter = timer1;


    OPEN_OR_CLOSE_DOOR();
  }
```

```
    /* If one minute has passed capture the measured sum during the last minute and start
sending the data */
   if(tempcounter2 > 600)
   {
     if(tempcounter2 > 600 && tempcounter2 < 620 && loopstate == 0)
     {
       flow_measure_total = flow_measure_sum/_PROMEDIO;
       Serial.print (flow_measure_total, DEC);
       Serial.print (" L/min\r\n");


       loopstate = 1;
     }
     send_data();
   }
     break;


   case 0:
     delay(100);
     SIM900.println("AT+CMGF=1");
     delay(100);
     SIM900.println("AT+CMGS=\"8298636916\"");
     delay(100);
     SIM900.println("INTRUSO DETECTADO FAVOR VISITAR LA CASA DE IVAN");
     delay(100);
     SIM900.write((char)26);
     delay(100);
     variable_button = 3;
     Serial.print("INTRUSO DETECTADO\r\n");
     default:
     break;
  }
```

}