Comunicacion de la placa con el módulo SIM800L enviando comandos AT.

AT – Es el comando AT más básico. También inicializa el Auto-baud'er. Si funciona, deberías ver el eco de los caracteres AT y luego OK, diciéndote que está bien y que te está entendiendo correctamente. Puedes enviar algunos comandos para consultar el módulo y obtener información sobre él, como por ejemplo

AT+CSQ – Comprueba la "fuerza de la señal" – el primer # es la fuerza en dB, debería ser mayor que alrededor de 5. Más alto es mejor. Por supuesto que depende de tu antena y de tu ubicación.

AT+CCID – Obtener el número de la tarjeta SIM – esto prueba que la tarjeta SIM se encuentra bien y puede verificar que el número está escrito en la tarjeta.

AT+CREG? Comprueba que estás registrado en la red. El segundo número debe ser 1 o 5. 1 indica que estás registrado en la red doméstica y 5 indica que estás en la red de roaming. Aparte de estos dos números indica que no estás registrado en ninguna red.

```
mySerial.println("AT"); //Once the handshake test is successful, it will back to OK

updateSerial();

mySerial.println("AT+CSQ"); //Signal quality test, value range is 0-31 , 31 is the best

updateSerial();

mySerial.println("AT+CCID"); //Read SIM information to confirm whether the SIM is plugged

updateSerial();

mySerial.println("AT+CREG?"); //Check whether it has registered in the network

updateSerial();
```

En la parte de bucle del código, llamamos a la función personalizada llamada updateSerial() que espera continuamente cualquier entrada del monitor serie y la envía al módulo SIM800L a través del pin D2 (Rx del módulo). También lee continuamente el pin D3 (Tx del módulo) si el módulo SIM800L tiene alguna respuesta.

```
void updateSerial()

{

delay(500);
```

```
while (Serial.available())
{
mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port
}
while(mySerial.available())
{
Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port
}
}
```

Deberías ver abajo la salida en el monitor serie.

```
delay(1000);

mySerial.println("AT"); //Once the handshake test is successful, it will back to OK
updateSerial();

mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
updateSerial();
mySerial.println("AT+CMGS="+ZZxxxxxxxxxx"");//change ZZ with country code and xxxxxxxxxxx with phone number to sms
updateSerial();
mySerial.print("Last Minute Engineers | lastminuteengineers.com"); //text content
updateSerial();
mySerial.write(26);
```

```
}


void loop()

{

}


void updateSerial()

{

delay(500);

while (Serial.available())

{

mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port

}

while(mySerial.available())

{

Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port

}

}
```

El código es casi igual al anterior, excepto por el fragmento de código. Una vez que se establece la conexión, enviamos los comandos AT de abajo:

AT+CMGF=1 – Selecciona el formato del mensaje SMS como texto. El formato por defecto es Unidad de Datos de Protocolo (PDU)

AT+CMGS=+ZZxxxxxxxxxx – Envía un SMS al número de teléfono especificado. El mensaje de texto introducido seguido de un carácter “Ctrl+z” se trata como un SMS. “Ctrl+z” es en realidad un 26º carácter no impreso descrito como “sustituto” en la tabla ASCII. Por lo tanto, necesitamos enviar 26DEC (1AHEX) una vez que enviemos un mensaje.

```
mySerial.println("AT+CMGF=1"); // Configuring TEXT mode

updateSerial();

mySerial.println("AT+CMGS="+ZZxxxxxxxxxx"");//change ZZ with country code and xxxxxxxxxxx with phone number to sms

updateSerial();

mySerial.print("Last Minute Engineers | lastminuteengineers.com"); //text content

updateSerial();

mySerial.write(26);
```

El bucle se mantiene vacío ya que queremos enviar un SMS sólo una vez. Si deseas enviar un SMS una vez más, sólo tienes que pulsar la tecla RESET en tu Arduino. Y entonces enviará al mensaje al número de teléfono designado.

Código de Arduino para Lectura de SMS

Ahora programemos nuestro Arduino para leer los mensajes entrantes. Este sketch es muy útil cuando se necesita activar una acción cuando se recibe un SMS específico. Por ejemplo, cuando el Arduino recibe un SMS, puedes darle instrucciones para activar o desactivar un relé.

```
#include

//Create software serial object to communicate with SIM800L

SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 & #2


void setup()

{

//Begin serial communication with Arduino and Arduino IDE (Serial Monitor)

Serial.begin(9600);

//Begin serial communication with Arduino and SIM800L

mySerial.begin(9600);

Serial.println("Initializing...");

delay(1000);
```

```
mySerial.println("AT"); //Once the handshake test is successful, it will back to OK

updateSerial();

mySerial.println("AT+CMGF=1"); // Configuring TEXT mode

updateSerial();

mySerial.println("AT+CNMI=1,2,0,0,0"); // Decides how newly arrived SMS messages should be handled

updateSerial();

}


void loop()

{

updateSerial();

}


void updateSerial()

{

delay(500);

while (Serial.available())

{

mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port

}

while(mySerial.available())

{

Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port

}

}
```

El código es similar al anterior, excepto por el fragmento de código de abajo. Una vez que se establece la conexión, enviamos abajo los comandos AT:

AT+CMGF=1 – Selecciona el formato del mensaje SMS como texto. El formato por defecto es Unidad de Datos de Protocolo (PDU)

AT+CNMI=1,2,0,0,0 – especifica cómo se deben manejar los mensajes SMS recién llegados. De esta manera, puede indicar al módulo SIM800L que reenvíe los mensajes SMS recién llegados directamente al PC o que los guarde en el almacenamiento de mensajes y luego notifique al PC sobre su ubicación en el almacenamiento de mensajes.

Su respuesta comienza con +CMT: Todos los campos de la respuesta están separados por comas y el primer campo es el número de teléfono. El segundo campo es el nombre de la persona que envía el SMS. El tercer campo es una marca de tiempo mientras que el cuarto campo es el mensaje real.

```
mySerial.println("AT+CMGF=1"); // Configuring TEXT mode

updateSerial();

mySerial.println("AT+CNMI=1,2,0,0,0"); // Decides how newly arrived SMS messages should be handled

updateSerial();
```

buscamos mensajes SMS recién llegados. Una vez que envíe el SMS al módulo GSM SIM800L, verá la salida en el monitor serial.

El búfer de recepción de SoftwareSerial se está llenando y descartando caracteres. No estás leyendo lo suficientemente rápido del buffer.

La solución más simple para esto es aumentar el tamaño del búfer del SoftwareSerial de su tamaño predeterminado de 64 bytes a 256

Guia solucion: En un PC con Windows, ve a C:Ficheros de programa (x86) -> Arduino -> hardware -> Arduino -> avr -> bibliotecas -> SoftwareSerial (-> src para la nueva versión del IDE de Arduino) Abre SoftwareSerial.h y cambie la línea:

```
// RX buffer size
```

#define _SS_MAX_RX_BUFF 64

a

// RX buffer size

#define _SS_MAX_RX_BUFF 256

Guarda el archivo y vuelve a ejecutar el código.

Código para hacer una llamada

Ahora programemos a nuestra placa para que haga la llamada. Este sketch es muy útil cuando quieres que haga una llamada para encender la luz del led tambien puede ser de SOS en caso de emergencia como que se exceda la temperatura o alguien entre en tu casa depende para que se utilize.

Antes de probar el código, tienes que introducir el número de teléfono. Busca la cadena ZZxxxxxxxxxx y sustituye ZZ por el código del condado y xxxxxxxxxx por el número de teléfono de 10 dígitos.

#include

//Create software serial object to communicate with SIM800L

SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 & #2

void setup()

{

//Begin serial communication with Arduino and Arduino IDE (Serial Monitor)

Serial.begin(9600);

//Begin serial communication with Arduino and SIM800L

mySerial.begin(9600);

Serial.println("Initializing...");

delay(1000);

```
mySerial.println("AT"); //Once the handshake test is successful, i t will back to OK

updateSerial();


mySerial.println("ATD+ +ZZxxxxxxxxxx;"); // change ZZ with country code and xxxxxxxxxxx
with phone number to dial

updateSerial();

delay(20000); // wait for 20 seconds...

mySerial.println("ATH"); //hang up

updateSerial();

}


void loop()

{

}


void updateSerial()

{

delay(500);

while (Serial.available())

{

mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port

}

while(mySerial.available())

{

Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port
```

}

}

Para realizar una llamada se utilizan los comandos AT:

ATD+ +ZZxxxxxxxxxx; – Marca un número determinado. El modificador de punto y coma (;) al final separa la cadena de marcado en múltiples comandos de marcado. Todos los comandos excepto el último deben terminar con el modificador punto y coma (;).

ATH – Cuelga la llamada

mySerial.println("ATD+ +ZZxxxxxxxxxx;"); // change ZZ with country code and xxxxxxxxxxx with phone number to dial

updateSerial();

delay(20000); // wait for 20 seconds...

mySerial.println("ATH"); //hang up

updateSerial();

Código de la placa para recibir la llamada

Recibir una llamada no requiere ningún código especial, sólo tienes que seguir escuchando el módulo SIM800L. Sin embargo, puede que encuentres este esquema muy útil, cuando necesites desencadenar una acción cuando se reciba una llamada de un número de teléfono específico.

#include

//Create software serial object to communicate with SIM800L

SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 & #2

void setup()

```
{
//Begin serial communication with Arduino and Arduino IDE (Serial Monitor)
Serial.begin(9600);

//Begin serial communication with Arduino and SIM800L
mySerial.begin(9600);

Serial.println("Initializing...");
}

void loop()
{
updateSerial();
}

void updateSerial()
{
delay(500);
while (Serial.available())
{
mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port
}
while(mySerial.available())
{
Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port
}
```

}

La llamada entrante suele indicarse con un “ANILLO” en el monitor de serie seguido del número de teléfono y el identificador de llamadas. Para aceptar o colgar una llamada se utilizan los siguientes comandos AT:

ATA – Acepta la llamada entrante.

ATH – Cuelga la llamada. Al colgar la llamada envía NO CARRIER en el monitor serial indicando que la llamada no pudo conectarse.

Esto se mostrara en el monitor serial como una llamada recibida por el módulo SIM800L GSM.