

**Universidad de las Américas Puebla**

**Decanato de Ingeniería**

**Departamento de Computación, Electrónica y Mecatrónica**

**Laboratorio de Sistemas de Telecomunicaciones**

Práctica 6

# **Aplicación de telefonía GSM con Arduino**

## **Integrantes:**

Carlos Andrés Reyes Evangelista	157068
Erick Siordia Nagaya	157504



# Práctica 6: Aplicación de telefonía GSM con Arduino

---

## Objetivo

Implementar un sistema de monitoreo de eventos con alerta de eventos via SMS y comunicación celular.

## Material

- GSM shield para Arduino.
- Arduino UNO
- Cable de conexión USB para Arduino
- SIM telefónico
- Adaptador de Micro/Nano SIM a MiniSIM
- Computador personal con software Arduino

## Resumen

Para lograr el objetivo de la práctica, se usó un Arduino Uno y un GSM Shield compatible con un SIM activo. El Arduino se conectó a una PC con el software de Arduino instalado. Con ayuda de este, se pudo compilar, cargar y ejecutar un programa que permitiera mandar un mensaje SMS a un número de teléfono.

Posteriormente, se utilizó un *protoboard* y un botón para poder crear un programa que reaccione a la entrada del usuario. Gracias a estos experimentos, se pudo utilizar el código analizado para crear un programa que mande un mensaje SMS a un número predeterminado cada vez que el botón conectado sea presionado. Gracias a esta práctica se pudo comprender de mejor manera el uso que se le puede dar a un Arduino para utilizar sistemas de telecomunicaciones como la telefonía celular; específicamente los mensajes SMS.

## Marco teórico

La tecnología GSM (Global System for Mobile Communications) fue desarrollada a mediados de los 80's, y cambió de gran manera la forma en que las personas se comunicaban. Al año de 2010, 3 billones de personas hacían uso de esta tecnología, y a pesar de haber nuevos avances como LTE, las redes GSM siguen siendo muy populares debido a su precio y a las constantes mejoras que se han hecho a su diseño a través de los años.

"Muchos operadores continúan invirtiendo en sus redes GSM en adición a sus actividades UMTS y LTE para introducir nuevas funcionalidades y para bajar los costos operacionales" (Sauter, 2011)

Fue en 1982 que se creó este estándar común para la comunicación celular digital en Europa. En éste se especificó que el sistema operaría sobre la frecuencia de 900 MHz. La primera llamada sobre GSM se hizo en 1991 y fue hacia el primer ministro de Finlandia. Ese mismo año, el estándar GSM fue expandido para ocupar también la frecuencia de 1800 MHz.

Para lograr el funcionamiento de un sistema GSM, se necesita implementar 3 componentes:

- Estación móvil
- Estación base
- Sub-sistema de conmutación de red

La estación móvil es el componente que interactúa con el usuario final. Específicamente se trata de un teléfono celular y de un chip SIM (*Subscriber Identity Module*), el cual permite al usuario ser identificado por el sistema.

La estación base es donde todas las funciones relacionadas con la radio son realizadas. Es a su vez la encargada de proveer sistemas de control para conectar los dispositivos móviles con el centro de conmutación de red.

Por otro lado, el sistema de conmutación de red es responsable de procesar las llamadas que los usuarios realizan y reciben gracias a un centro de servicio de conmutación de red, como también es encargado de manejar datos como la información de los subscriptores.

La frecuencia portadora de las señales GSM varía según las regiones del mundo; en Norteamérica las frecuencias 850 y 1900 MHz son usadas, mientras que en otros países se usan como se mencionó antes las frecuencias 900 o 1800 MHz. Igualmente, las frecuencias de 400 y 450 MHz también son usadas para GSM en algunos lugares para aprovechar el espectro de frecuencias del sistema de la generación previa. Finalmente, en Europa también se ocupa la banda de 2100 MHz para la transmisión de 3G en GSM.

## Desarrollo de la práctica

### Configuración de la placa *Arduino*

En primer lugar, fue conectado el *Shield GSM* a la placa de *Arduino UNO* conectando cada puerto del *Arduino* contra cada pin en el *Shield GSM* en el cual se conectó una tarjeta SIM. Esta configuración es ilustrada en la *Figura 1*.

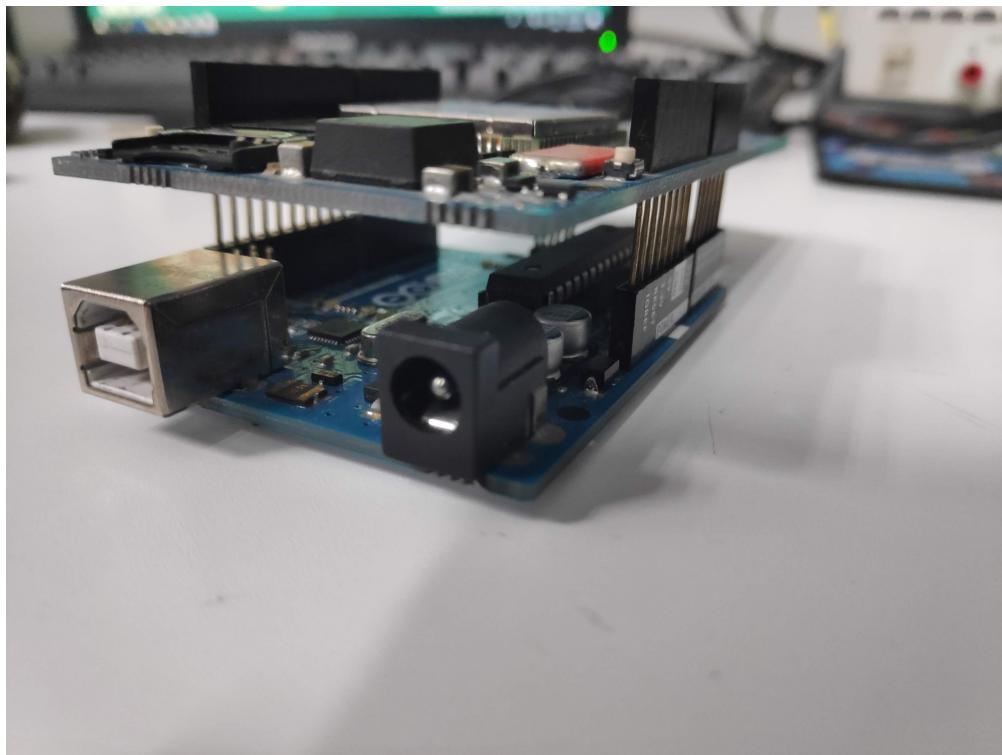


Figura 1. Placa de *Arduino* conectada al *Shield GSM*

Finalmente, se conectaron las entradas de corriente del botón al puerto número 4 del *Shield GSM* y a tierra, correspondientemente, tal como se muestra en la *Figura 2*.

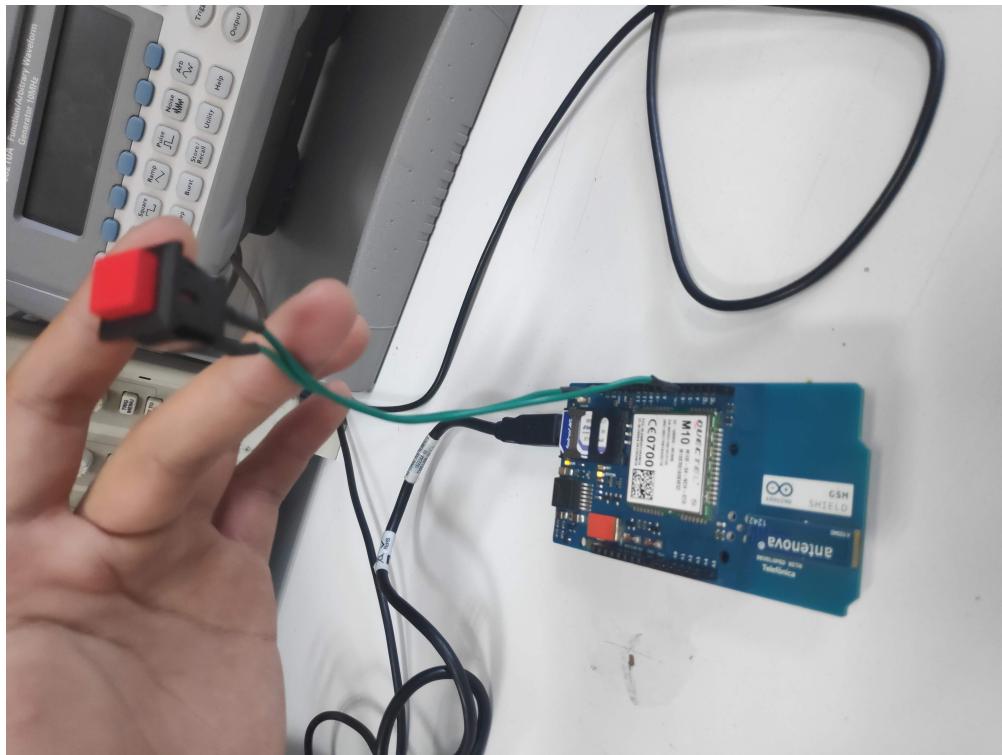


Figura 2. Botón de entrada conectado al Shield GSM. Puerto 4 y tierra.

El Arduino finalmente es conectado vía USB a la computadora y se selecciona el puerto en el que éste esté operando.

Diseñe e implemente un programa para Arduino UNO que sea capaz de enviar un mensaje SMS a un número telefónico predeterminado

Para realizar esta tarea fueron consultados los bosquejos de ejemplo que el mismo software de [Arduino](#) provee. El código utilizado que se encarga del envío de mensajes es el siguiente:

```
// Include the GSM library
#include <GSM.h>

#define PINNUMBER ""

// initialize the library instance
GSM gsmAccess;
GSM_SMS sms;

void setup() {
    // initialize serial communications and wait for port to open:
    Serial.begin(9600);
    while (!Serial) {
        ; // wait for serial port to connect. Needed for native USB port only
    }

    Serial.println("SMS Messages Sender");

    // connection state
    boolean notConnected = true;
```

```

// Start GSM shield
// If your SIM has PIN, pass it as a parameter of begin() in quotes
while (notConnected) {
    if (gsmAccess.begin(PINNUMBER) == GSM_READY) {
        notConnected = false;
    } else {
        Serial.println("Not connected");
        delay(1000);
    }
}

Serial.println("GSM initialized");
}

void loop() {
    Serial.print("Enter a mobile number: ");
    char remoteNum[20]; // telephone number to send sms
    readSerial(remoteNum);
    Serial.println(remoteNum);

    // sms text
    Serial.print("Now, enter SMS content: ");
    char txtMsg[200];
    readSerial(txtMsg);
    Serial.println("SENDING");
    Serial.println();
    Serial.println("Message:");
    Serial.println(txtMsg);

    // send the message
    sms.beginSMS(remoteNum);
    sms.print(txtMsg);
    sms.endSMS();
    Serial.println("\nCOMPLETE!\n");
}

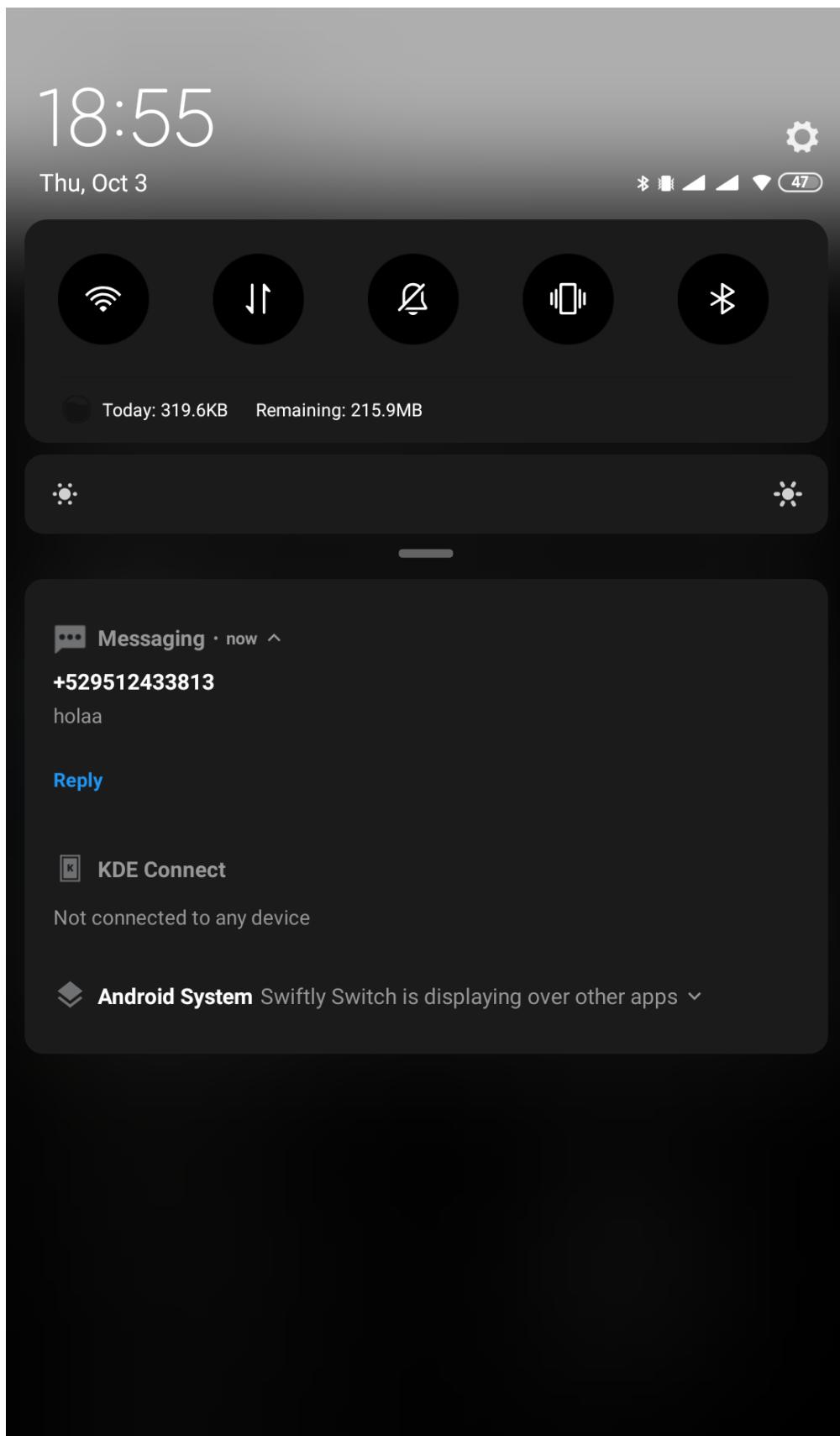
/*
Read input serial
*/
int readSerial(char result[]) {
    int i = 0;
    while (1) {
        while (Serial.available() > 0) {
            char inChar = Serial.read();
            if (inChar == '\n') {
                result[i] = '\0';
                Serial.flush();
                return 0;
            }
            if (inChar != '\r') {
                result[i] = inChar;
                i++;
            }
        }
    }
}

```

```
}
```

Cargue y ejecute el programa en el Arduino

En cuanto el programa es cargado en el Arduino y se ingresa un número telefónico y un cuerpo de mensaje, éste es enviado y el receptor lo recibe inmediatamente, como puede observarse en la Figura 3.



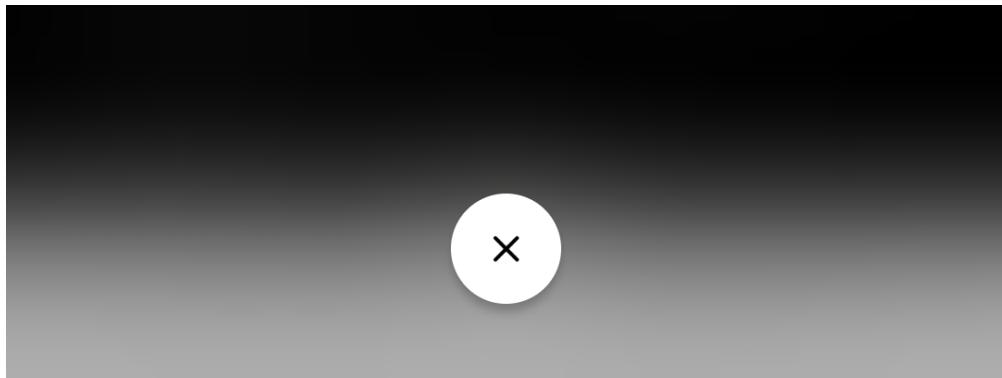


Figura 3. Mensaje recibido desde el programa cargado al Arduino

Modifique el programa para que al activar un botón envíe un mensaje

Después de comprobar que el envío de mensajes funciona exitosamente, fue necesario modificar el programa de tal manera que en cuanto sea detectado un clic en el botón conectado al Shield GSM se envíe un mensaje predefinido a un número predeterminado. Una vez que se detectó el código necesario para evaluar el estado del botón en tiempo real, únicamente fue necesario combinar ambas funcionalidades para conseguir el objetivo previsto. El código final del módulo encargado de evaluar el estado del botón y enviar el mensaje en cuanto se detecte que éste fue presionado luce como sigue:

```
#include <GSM.h>

GSM gsmAccess;
GSM_SMS sms;

#define PINNUMBER ""

void setup() {
    //start serial connection
    Serial.begin(9600);
    //configure pin 4 as an input and enable the internal pull-up resistor
    pinMode(4, INPUT_PULLUP);
    pinMode(13, OUTPUT);

    while (!Serial) {
        ; // wait for serial port to connect. Needed for native USB port only
    }

    Serial.println("SMS Messages Sender");

    // connection state
    boolean notConnected = true;

    // Start GSM shield
    // If your SIM has PIN, pass it as a parameter of begin() in quotes
    while (notConnected) {
        if (gsmAccess.begin(PINNUMBER) == GSM_READY) {
            notConnected = false;
        } else {
            Serial.println("Not connected");
            delay(1000);
        }
    }
}
```

```

    }

}

Serial.println("GSM initialized");

}

void loop() {
    //print out the value of the pushbutton
    int sensorVal = digitalRead(4);
    Serial.println(sensorVal);

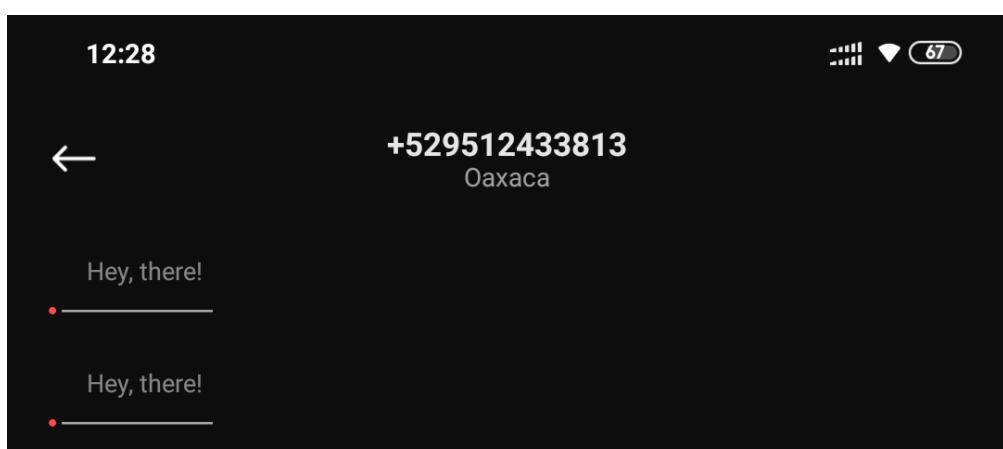
    char remoteNum[20] = "2227209279"; // telephone number to send sms
    // readSerial(remoteNum);

    char txtMsg[200] = "Hey, there!";
    // readSerial(txtMsg);

    // Keep in mind the pull-up means the pushbutton's logic is inverted. It goes
    // HIGH when it's open, and LOW when it's pressed. Turn on pin 13 when the
    // button's pressed, and off when it's not:
    if (sensorVal == HIGH) {
        sms.beginSMS(remoteNum);
        sms.print(txtMsg);
        sms.endSMS();
        Serial.println("\nCOMPLETE!\n");
        digitalWrite(13, LOW);
    } else {
        digitalWrite(13, HIGH);
    }
}

```

Una vez el código fue compilado y cargado a la placa Arduino, fue realizada una prueba presionando el botón para revisar su funcionamiento. Durante esa prueba, fue percibido el hecho de que el ciclo en que la condicional corre es ejecutado a cada muestra en unas cuantas fracciones de segundo, lo que significa que incluso una corta presión en el botón enviará multitud de mensajes. Este comportamiento es mostrado en la Figura 4, donde de un par de presiones se registraron hasta 84 mensajes.



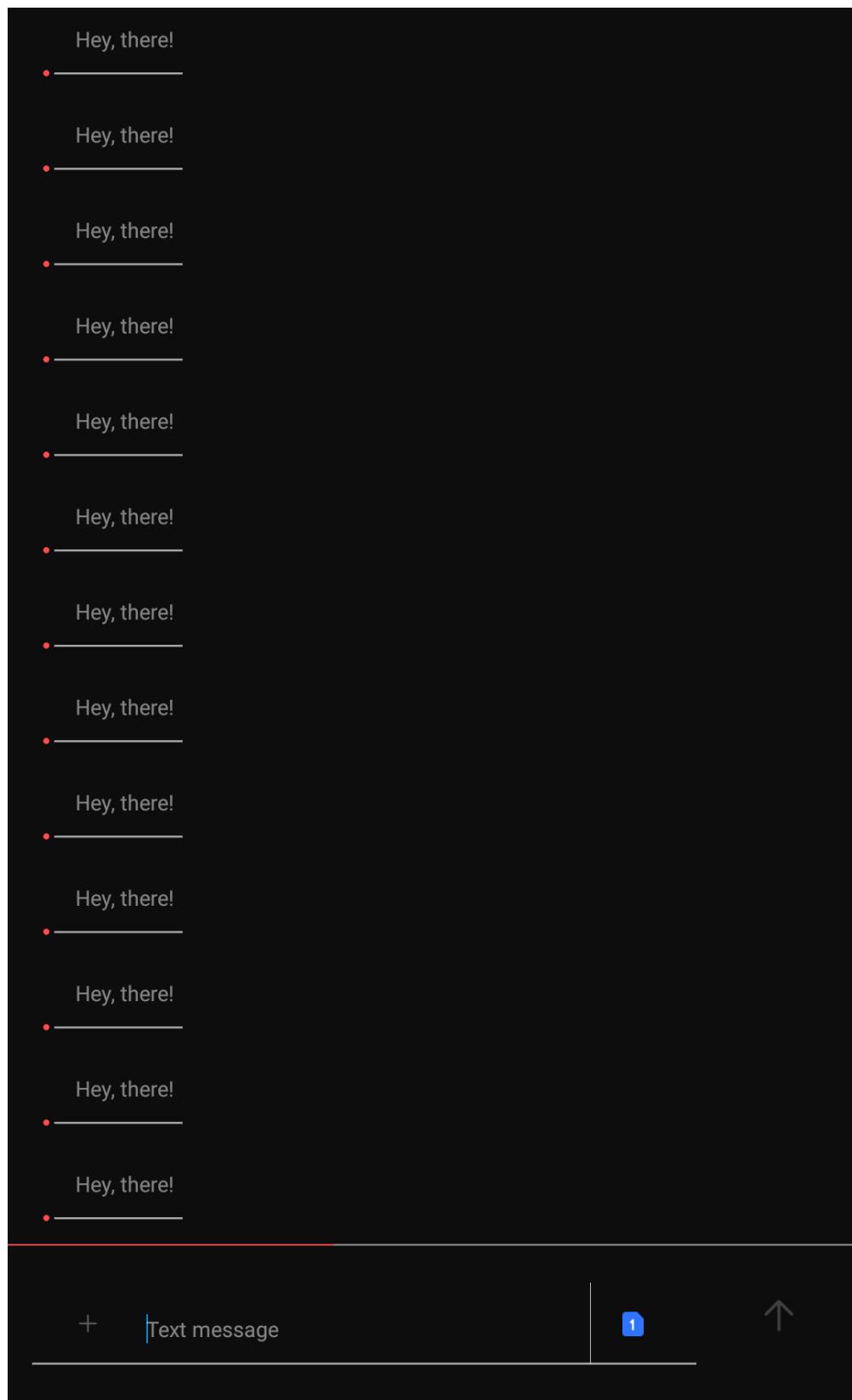


Figura 4. Mensajes recibidos desde el programa cargado al Arduino

## Conclusión

Gracias a la presente práctica se pudo comprender de una mejor manera el uso que los dispositivos como los Arduino pueden tener en el área de telecomunicaciones. Igualmente, fue gracias a esta que se pudo tener un primer acercamiento a la programación, configuración y uso del software Arduino en una PC para programar rutinas que un Arduino pueda llevar a cabo. Además, el desarrollo de este experimento e investigación permitió una asimilación más profunda sobre el concepto y aplicación de la red GSM.

## Referencias

- Sauter, M. (2011). From GSM to LTE-Advanced. John Wiley and Sons.
- F.U. Nweke, S.I. Anyigor and A.E. Umahi. Global System for Mobile Communication, GSM Network Operation Call Rate in Nigeria. (2019). Middle-East Journal of Scientific Research, [online] 23(9), pp.2343-2346. Available at: [https://www.idosi.org/mejsr/mejsr23\(9\)15/47.pdf](https://www.idosi.org/mejsr/mejsr23(9)15/47.pdf) [Accessed 9 Oct. 2019].