

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**



**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y  
COMPUTACIÓN**

**“DISEÑO E IMPLEMENTACION DE PROTOTIPO PARA  
CONTROL DE RASPBERRY PI MEDIANTE MENSAJES GSM  
EN APLICACIONES DE SEGURIDAD”**

**TESINA DE SEMINARIO**

Previa la obtención del Título de:

**INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES**

**Presentado por:**

**IRWIN DIONICIO RODRÍGUEZ DÍAZ  
JIMMY OLMEDO MEDINA COLLAHUAZO**

**GUAYAQUIL – ECUADOR**

**AÑO 2013**

## **AGRADECIMIENTO**

En primer lugar este agradecimiento es a Dios por permitirme a cumplir este sueño segundo lugar a mis padres por ser un constante apoyo y confiar en mí, y por ultimo agradecemos a nuestros profesores y compañeros que nos han ayudado a culminar con éxito esta etapa tan importante de mi vida.

Irwin Rodríguez

En primer lugar le agradezco a Dios por permitirme terminar con éxito este trabajo, y a mis padres por el apoyo constante que a diaria eh recibido, que han sido un pilar fundamental en toda mi vida.

Jimmy Medina

## DEDICATORIA

Esta obra va dedicada a mis padres ya que por sus valores inculcados son mi fuente de inspiración aun si apoyo y confianza no hubiera sido posible convertirme en la persona que soy hoy.

Irwin Rodríguez

Este trabajo se lo dedico a mis padres por todo el esfuerzo que mis padres me han ofrecido, el apoyo y la confianza que me han brindado se ve reflejada en la persona que ahora soy.

Medina Jimmy

## **TRIBUNAL DE SUSTENTACIÓN**

---

Ing. Carlos Valdivieso A.

**PROFESOR DEL SEMINARIO DE GRADUACIÓN**

---

Ing. Hugo Villavicencio V.

**PROFESOR DELEGADO DE LA UNIDAD ACADÉMICA**

## **DECLARACIÓN EXPRESA**

“La responsabilidad del contenido de esta Tesina, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral”.

(Reglamento de Graduación de la ESPOL)

---

**Medina Collahuazo Jimmy**

---

**Rodríguez Díaz Irwin**

## RESUMEN

El siguiente trabajo de estudio académico detalla varios puntos importantes de nuestro proyecto como son investigación, análisis, implementación, adquisición y procesamiento de datos, además de todos los conocimientos adquiridos en nuestra carrera como estudiantes politécnicos, enfocándonos en el campo de los microcontroladores. Puntualizaremos el estudio y correcto funcionamiento de la tarjeta Raspberry Pi, denominada como la computadora de bolsillo debido a su tamaño, semejante a una tarjeta de identificación personal, además del uso de los módulos arduino shield y el módulo GPRS/GSM QUADBAND (SIM900) .

El proyecto consiste en una alarma de seguridad que al ser encendida manda la señal a nuestra raspberry pi y con la programación ejecutada activa la interfaces del arduino shield con el módulo GPRS/GSM QUADBAND (SIM900), este módulo es muy importante ya que la función es de mandar un mensaje de texto en la cual nos indica que se activó la alarma, para hacer posible este programa utilizamos la librería ardupi y el programa cutecom que nos indica que el módulo ya está encendido y podemos hacer pruebas de envíos de mensaje.

## ÍNDICE GENERAL

AGRADECIMIENTO.....	i
DEDICATORIA.....	ii
TRIBUNAL DE SUSTENTACION.....	iii
DECLARACION EXPRESA.....	iv
RESUMEN.....	v
ÍNDICE GENERAL.....	vi
ÍNDICE ABREVIATURA.....	xi
ÍNDICE DE FIGURAS.....	xii
INTRODUCCION.....	xvi
<b>CAPITULO 1.....</b>	<b>1</b>
<b>DESCRIPCIÓN GENERAL DEL PROYECTO.....</b>	<b>1</b>
1.1.1 ANTECEDENTES.....	1
1.1.2 OBJETIVO.....	2
OBJETIVO GENERAL.....	2
OBJETIVO ESPECÍFICOS.....	2
1.3 IDENTIFICACION DEL PROBLEMA.....	3
1.4 LIMITACIONES.....	4
<b>CAPITULO 2.....</b>	<b>5</b>
<b>FUNDAMENTO TEORICO.....</b>	<b>5</b>
2.1 RESUMEN.....	5

2.2 RASPBERRY PI.....	6
2.3 CARACTERISTICA.....	8
2.3.1 PROCESADOR Y MEMORIA.....	8
2.3.2 CARACTERISTICA TECNICAS DE LA GPU.....	9
2.3.3 CONECTORES.....	9
2.3.4 ENTRADAS USB 2.0.....	9
2.3.5 SALIDA DE VIDEO DIGITAL HDMI Y ANALÓGICO.....	10
2.3.6 GPIO EN LA RASPBERRY PI.....	12
2.3.7 ALIMENTACION.....	13
2.3.8 DISPOSITIVO DE ARRANQUE.....	13
2.4 SOFTWARE.....	15
2.5 SISTEMA OPERATIVO SOPORTADOS.....	15
2.5.1 SISTEMA OPERATIVO RASPBIAN WHEEZY.....	15
2.6 RANGO DE TEMPERATURA.....	16
2.7 DIMENSIONES.....	16
2.8 ARDUINO SHIELD.....	17
2.8.1 ARDUINOSHIELD Y SUS COMPONENTES.....	18
2.9 HERRAMIENTA DE PROGRAMACION.....	21
2.9.1 SISTEMA OPERATIVO LINUX.....	21
2.9.2 SISTEMA OPERATIVO DEBIAN.....	22
2.9.3 PROGRAMACION EN PYTHON.....	23
2.10 MODULO GPRS/GSM QUADBAND (SIM900).....	24



2.10.1 CARACTERISTICA.....	25
2.10.2 FUENTE DE ALIMENTACION.....	26
2.10.3 ESQUEMÁTICO DEL GSM.....	28
<b>CAPITULO 3</b>	
DESARROLLO DEL PROYECTO.....	31
3.1 RESUMEN.....	31
3.2 EJERCICIO 1 INTERFAZ DE RASPBERRY PI CON EL ARDUINO SHIELD.....	33
3.2.1 DIAGRAMA DE BLOQUE.....	33
3.2.2 CÓDIGO FUENTE.....	34
3.2.3 CONCLUSIÓN.....	35
3.3 EJERCICIO 2 INTERFAZ ENTRE GPRS/GSM QUADBAND (SIM900) ARDUINO SHIELD Y RASPBERRY PI.....	35
3.3.1 DIAGRAMA DE BLOQUE.....	36
3.3.2 CÓDIGO FUENTE.....	36
3.3.3 CONCLUSION.....	38
3.4 EJERCICIO 3 PROBANDO EL MÓDULO GPRS/GSM QUADBAND (SIM900) CON EL PROGRAMA CUTECOM.....	39
3.4.1 DIAGRAMA DE BLOQUE.....	39
3.4.2 CONCLUSION.....	39
3.5 EJERCICIO 4 INTERFAZ ENTRE EL MÓDULO GPRS/GSM Y LA RASPBERRY PI.....	40

3.5.1 DIAGRAMA DE BLOQUE.....	40
3.5.2 CÓDIGO FUENTE.....	40
3.5.3 CONCLUSION.....	43
3.6 PROYECTO TERMINADO.....	43
3.6.1 DIAGRAMA DE BLOQUES.....	43
3.6.2 DIAGRAMA DE FLUJO.....	44
3.6.3CÓDIGO FUENTE.....	46
3.6.4 CONCLUSION.....	51
<b>CAPITULO 4</b>	
SIMULACIONES Y PRUEBAS EXPERIMENTADAS.....	52
4.1 RESUMEN.....	52
4.2 IMÁGENES DEL EJERCICIO 1.....	53
4.3SIMULACION EJERCICIO 2.....	55
4.4 SIMULACION EJERCICIO 3.....	57
4.5 SIMULACION EJERCICIO 4.....	59
4.6 SIMULACION PROYECTO.....	61
CONCLUSIONES.....	63
RECOMENDACIONES.....	65
BIBLIOGRAFIA.....	66
ANEXO 1.....	69
ESQUEMÁTICO MÓDULO GPRS/GSMQUADBAND (SIM900).....	69
ANEXO 2.....	70

ESQUEMÁTICO DEL ARDUINO SHIELD.....	70
ANEXO 3.....	71
GRÁFICO DE LOS PINES DE LAS RASPEBRRY PI.....	71
ANEXO 4.....	71
MENÚ INICIAL DE CONFIGURACIÓN DE LA RASPBERRY PI.....	71
ACTIVACION DE PUERTO UART.....	72
COMPILAR BIBLIOTECA ARDUPI.....	72
EJECUCIÓN DEL PROGRAMA.....	73

## ÍNDICE DE ABREVIATURAS

**DSI** (DISPLAY SERIAL INTERFACE)

**IDE** (INTEGRADED DEVELOPMENT ENVIRONMENT)

**GPIO** (GENERAL PURPOSE INPUT OUTPUT)

**GPS** (GLOBAL POSITIONING SYSTEM)

**GPU** (GRAPHICS PROCESSING UNIT)

**HDMI** (HIGH DEFINITION MULTIMEDIA INTERFACE)

**LED** (LIGHT-EMITTING DIODE)

**PWM** (PULSE WIDTH MODULATION)

**RAM** (RANDOM ACCESS MEMORY)

**RX** (RECEIVER)

**SD** (SECURE DIGITAL)

**SPI** (SERIAL PERIPHERAL INTERFACE)

**TX** (TRANSMITTER)

**URL** (UNIFORM RESOURCE LOCATOR)

**USB** (UNIVERSAL SERIAL BUS)

## ÍNDICE DE FIGURAS

Figura 2.2.Raspberry pi y sus componentes.....	6
Figura 2.3.Procesador y memoria de la raspberry pi.....	8
Figura 2.3.4.Puertos de usb y entrada Ethernet.....	10
Figura 2.3.5. (a).Puertos de salida HDMI.....	11
Figura 2.3.5 (b).Puertos de salida RCA.....	12
Figura 2.3.7.Adaptador de alimentación de la raspberry pi.....	13
Figura 2.3.8.Memoria sd card.....	14
Figura 2.8.Arduinoshield.....	17
Figura 2.8.1.Arduinoshield y sus conectores.....	19
Figura 2.8.2.Conector del arduinoshield a raspberry pi.....	20
Figura 2.9.1.Sistema operativo Linux.....	21
Figura 2.9.3.Programación en Python.....	23
Figura 2.10.Módulo GPRS/GSM Quadband (SIM900).....	24
Figura 2.10.2.Circuitos de alimentación.....	27
Figura 2.10.3.Diagrama del módulo GPRS/GSMQuadband (SIM90).....	28
Figura 3.2.1.Diagrama de bloque ejercicio 1.....	33
Figura 3.3.1.Diagrama de bloque ejercicio 2.....	36
Figura 3.4.1.Diagrama de bloque ejercicio 3.....	39
Figura 3.5.1 diagrama de bloque ejercicio 4.....	40

Figura 3.6.1.Diagrama de bloque del proyecto terminado.....	43
Figura 3.6.2(a).Diagrama de flujo del proyecto terminado.....	44
Figura 3.6.2 (b).Diagrama de flujo del proyecto terminado.....	45
Figura 4.2 (a).Código del ejercicio 1.....	53
Figura 4.2 (b).Código compilado.....	53
Figura 4.2 (c).Ejecutando programa.....	54
Figura 4.2 (d).Imágenes del ejercicio terminado.....	54
Figura 4.3(a).Código del ejercicio2.....	55
Figura 4.3 (b).Diseño en protoboard.....	55
Figura 4.3 (c).Ejercicio Ejecutado.....	56
Figura 4.4(a).Detectando el módulo sim900 con comando AT=ok.....	56
Figura 4.4 (b).Verificación de transmisión del módulo sim900.....	58
Figura 4.4 (c).Muestra el mensaje recibido por medio del programa Cutecom.....	58
Figura 4.5(a).Código.....	59
Figura 4.5 (b).Compilación y ejecución del programa.....	59
Figura 4.6 (a).Compilación y ejecución del código final.....	60
Figura 4.6 (b).Mensaje de la ejecución del programa.....	61
Figura 4.6(c).Proyecto funcionando.....	61

## INTRODUCCIÓN

Este proyecto se basa en una alarma de seguridad, en la cual por medio de un mensaje de texto le indicara al propietario o el encargado que se ha activado la alarma, este aviso lo desarrolla mediante un sensor que manda la señal a las raspberry pi y por medio de una programación activaremos el módulo GPRS/GSM Quadband (SIM900) que dará la orden de enviar el mensaje de activación de alarma.

El primer capítulo daremos conocer una descripción general del proyecto, sus alcances, limitaciones y objetivos que son necesarios para concluir nuestro proyecto.

Se hace énfasis como esta tarjeta raspberry pi ha revolucionado en el mercado de las computadoras y los sistemas operativos que se necesitan para el funcionamiento y los diferente lenguajes para programar y ejecutar programas.

En el segundo capítulo se especifica las herramientas de software y hardware y las importantes características de la raspberry pi y los módulos usados en nuestro proyecto como: la raspberry pi, arduino shield, módulo GPRS/GSM Quadband (SIM900).

El tercer capítulo hablaremos el diseño la implementación del proyecto, los pasos que seguimos para concluir nuestro proyecto con ejercicios que fuimos desarrollando para lograr una interfaz de la raspberry pi con los módulos que usamos, como el módulo arduino shield y módulo GPRS/GSM Quadband (SIM900) mostraremos los código fuentes con sus respectivos diagrama de flujo.

En el cuarto y último capítulo se realizan las pruebas y simulaciones para la finalización de nuestro proyecto, la programación y ejecución de la interfaces raspberry pi y los módulos, y el resultado final de nuestro proyecto el aviso del sistema de alarma mediante un mensaje de texto gsm.

Finalmente redactaremos las conclusiones acerca de nuestro proyecto, además de las recomendaciones que se deben llevar a cabo para efecto del mismo, así como para cualquier posible falla presentada en implementación o configuración de los dispositivos. También añadiremos información adicional sobre el desarrollo del proyecto.



# **CAPÍTULO 1**

## **DESCRIPCIÓN GENERAL DEL PROYECTO**

### **1.1 ANTECEDENTES**

La tarjeta raspberry pi, es una tecnología nueva en la cual es usada para una gran variedad de aplicaciones en la actualidad moderna ya que es un mini computador y podemos realizar sin número de actividades en ella como una PC normal.

En la actualidad es muy usado en el campo de automatización y robótica ya que podemos controlar un sistema mediante esta tarjeta.

Es considerado como un mini ordenador fácil de llevarlo y es potente además trabaja con el sistema operativo Linux es usado mucho ya que se puede utilizar como un dispositivo multimedia ya que tiene una bastante potencia para reproducir videos de alta resolución.

En la actualidad hay sin número de sistemas de alarma con diferentes funciones y tamaños lo que se trata de lograr en la actualidad es un aviso inmediato y ser receptado por dispositivos que usemos a diario para recibir el aviso de la activación de dicha alarma y lograr tener control de aquella.

## **1.2 OBJETIVOS**

### **OBJETIVO GENERAL**

- Adquirir la destreza necesaria en el sistema operativo Linux y las compatibilidades con diversos programas que utilizaremos en nuestro proyecto como entradas y comandos necesarios.

### **OBJETIVOS ESPECÍFICO**

- Implementar un sistema de alarma o de activación
- Programar correctamente para la activación de los módulos
- Obtener conocimientos básico del sistema operativo Linux
- Instruirse en las conexiones de cada módulo respectivo
- Lograr realizar el sistema efectivo y funcional que permita la exactitud del aviso en este caso el mensaje de texto.

### 1.3 IDENTIFICACION DEL PROBLEMA

El presente trabajo, se trata sobre un sistema de aviso en este caso una alarma de seguridad, que nos alerta la violación o de activación mediante un mensaje de texto, en el cual sabremos que se ha activado el sistema de alarma.

El problema a solucionar consiste en el aviso eficaz y rápido de una alarma encendida en la cual nos avisara en un mensaje de texto utilizando la tarjeta

raspberry pi y un módulo gsm en la cual con una básica programación nos enviará el mensaje de aviso que el sistema de alarma se ha activado.

Este problema sería resuelto de una manera muy eficaz y segura ya que en el preciso momento que nos indica por medio mensaje que hubo la interrupción podríamos dar aviso a las autoridades del percance y así evitar un problema económico mayor, con este sistema lograremos evitar cualquier percance delincuencia

## **1.4 LIMITACIONES**

Es importante también tener en cuenta las limitaciones en cuanto a los equipos disponibles para la pruebas correspondiente y necesarias del proyecto ya que son equipos importados y de un precio un poco elevado por algún motivo se descompone uno de los equipos volver a importarlos retrasaría el trabajo de investigación de este proyecto.

El Raspebrry PI es un dispositivo muy complejo en el momento de hacer la programación, necesitamos toda la información necesaria en libros o los que encontramos en el internet.

## **CAPÍTULO 2**

### **FUNDAMENTO TEÓRICO**

#### **2.1 RESUMEN**

En este capítulo haremos una breve descripción teórica para poder entender los funcionamientos de los elementos utilizados en este proyecto, como nuestro proyecto consiste en la implementación de prototipo de control de Raspberry pi mediante mensaje gsm describiremos cada herramienta necesaria para lograr nuestro objetivo.

Indicaremos a detalle el funcionamiento de cada dispositivo aplicado en el proyecto , también se explicara como añadir fácilmente GSM a nuestro proyecto basado a Arduino-Celular ya que esto es usado como un adaptador de GSM a Raspberry pi usando también una antena para mejora la señal de envío, para no tener dificultad al recibir el mensaje de

alerta o de activación de la alarma , este proceso funcionara al mandar una señal de activación que la registrara nuestro sistema programado en nuestra tarjeta raspberry pi y mandara el mensaje correspondiente dado por la programación del proyecto

## 2.2 RASPBERRY PI

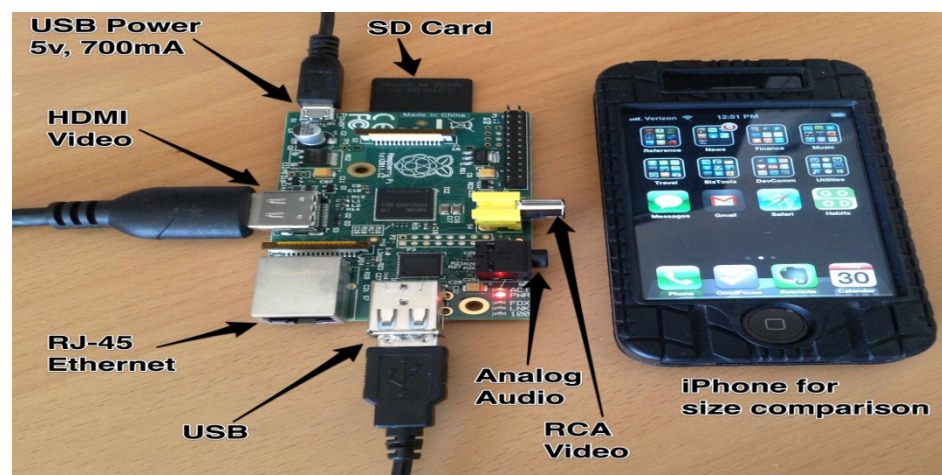


Figura 2.2. Raspberry pi y sus componentes [1]

La Raspberry Pi es un pequeño computador del tamaño de una tarjeta de crédito capaz de conectarse a tu TV o monitor y actuar como un fantástico reproductor multimedia, un mini servidor o todo lo que se te llegue a ocurrir que puedas hacer con Linux.

Si bien la fundación Raspberry Pi se ha hecho cargo principalmente de proporcionar el hardware del Modelo B, también ha ayudado para que la comunidad cree distintos sistemas operativos (todos basados en Linux)

que satisfagan distintas necesidades y soluciones con errores o problemas que tengamos al usarlo.

Ya que el sistema operativo se carga desde una simple tarjeta SD, tener varias distribuciones de Linux listas para ser cambiadas no entraña complicación alguna. De igual manera, es tan sencillo crear una distribución ARM para el Raspberry Pi que cada cierto tiempo se ven nuevas versiones de Linux en los foros del proyecto.

Ya que el Modelo B cuenta con dos puertos USB es sencillo conectarle un ratón y un teclado y usarlo como un ordenador cualquiera, aunque sea mucho más lento ordenadores con un precio más elevado. El problema es que si bien se puede usar una interfaz gráfica en el Pi, su limitada memoria y pequeño procesador hace que las cosas funcionen con bastante lentitud. Por eso debe quedar claro que los ordenadores de la fundación Raspberry Pi no fueron creados para remplazar a tu ordenador principal, sino como herramientas de aprendizaje y juego.

## 2.3 CARACTERÍSTICA

### 2.3.1 PROCESADOR Y MEMORIA

El Raspberry Pi modelo B, tiene una Broadcom BCM2835 es considerado la parte más importante de las raspberry , debido que se encarga de los procesos de audio, video, procesamiento gráfico y comunicación de las interfaces de hardware (teclado y mouse) como se aprecia en la figura 2.3.1, este sistema en un chip que incluye un ARM1176JZF-S 700 MHz del procesador (El firmware incluye una serie de "Turbo" Modos de modo que el usuario puede intentar overclocking, hasta 1 GHz, sin perjuicio de la garantía), con memoria de 256 MB.

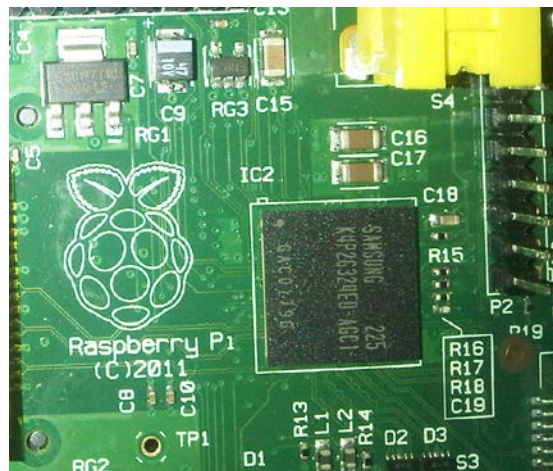


Figura 2.3.1. Procesador y memoria de la raspberry pi



### **2.3.2 CARACTERÍSTICAS TÉCNICAS DE LA GPU**

La GPU es capaz de mover contenidos con calidad Bluray, usando H.264 hasta 40MBits/s. Dispone un core 3D con soporte para las librerías OpenGL ES2.0 y OpenVG. Es capaz de decodificar 1080p30 H.264 high-profile.

### **2.3.3 CONECTORES**

- 2x entradas USB 2.0
- Entrada Ethernet RJ-45 10/100
- Salida de Video Digital HDMI (Audio y Video)
- Salida de Video Analógico (S-Video)
- Audio Analógico (Entrada 3,5mm)
- Entrada GPIO
- Entrada de alimentación Micro USB
- Lector de memorias SD (Utilizado para arrancar el dispositivo)

### **2.3.4 ENTRADAS USB 2.0**

EL Raspberry Pi posee 2 puertos USB, como se muestra en la figura 2.3.4, para la conexión de dispositivos como el teclado, mouse, impresoras, parlantes, todo dispositivos que sean compatible con las raspberry pi que tengan conectores USB, podemos adicionar un hub para

alimentar a más dispositivos con salida USB para así satisfacer nuestras necesidades necesaria al trabajar con esta maravillosa tarjeta.

Además el modelo actual del Raspberry Pi, posee un puerto Ethernet para conectarse al internet, como se aprecia en la figura 2.6, que la necesitaremos para actualizar y descargar programas necesarios para para el uso necesario para cada usuario.



Figura 2.3.4. Puertos de usb y entrada ethernet

### 2.3.5 Salida de Video Digital HDMI y Analógico

El Raspberry Pi está diseñado para soportar tres estándares diferentes de salida de video: RCA, HDMI, DSI. Los dos primeros son de fácil acceso al usuario ya que se encuentran en los televisores analógicos y digitales de la era actual, mientras que para el último estándar, es necesario un hardware adicional para su óptimo funcionamiento.

Para un televisor analógico es necesario utilizar la salida RCA, para que se muestra en la figura 2.3.5 (B), un televisor digital o bien puede seguir usando esta salida o en su defecto usar la salida HDMI que se muestra en la figura 2.3.5(A)



Figura 2.3.5(a).Puertos de salida HDMI



Figura 2.3.5 (b).Puertos de salida RCA

### 2.3.6 GPIO en la Raspberry Pi

El puerto de entrada/salida para propósitos generales (GPIO) son los pines que se aprecian en la figura 2.3.6, estos pines le permiten al Raspberry Pi comunicarse con otros componentes y circuitos electrónicos, además con una programación adecuada se dispone de la opción de censar temperatura, controlar servo motores y utilizando los protocolos adecuados como SPI e I2C comunicarse con otra computadora e incluso con otra Raspberry Pi. [6]

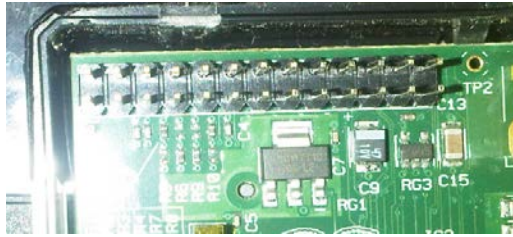


Figura 2.3.6. Puerto gpio de las raspberry pi

### 2.3.7 ALIMENTACIÓN

Vía Micro USB 5 Voltios. Casi cualquier dispositivo con alimentación USB puede servir como fuente de alimentación, como se muestra en la figura 2.3.7.



Figura 2.3.7. Adaptador de alimentación de la raspberry pi

### 2.3.8 DISPOSITIVO DE ARRANQUE

En la memoria SD tenemos descargado sistema operativo ocupa poco espacio en la memoria de la SD (2GB) debido a que apenas vienen instalados un mínimo de aplicaciones para el usuario. Esta memoria se puede decir que es el disco duro de la raspberry , ya que desde aquí arranca todos los programas y aplicaciones que se hayan descargado para la utilización de la raspberry pi la podemos observar en la figura 2.3.8.



Figura 2.3.8.Memoria sd card

## 2.4 SOFTWARE

El Raspberry Pi utiliza Linux basadas en el kernel sistemas operativos Raspbian , es un sistema basado en Debian operativo libre optimizado para el hardware Raspberry Pi, es el sistema actual recomendado.

La GPU de hardware se accede a través de un firmware imagen que se carga en la GPU en tiempo de arranque de la tarjeta SD.

## 2.5 SISTEMA OPERATIVO SOPORTADOS

- Raspbian “wheezy” (Debian)
- ArchLinux
- Fedora
- QtonPi (QT SDK)
- OpenElec
- Raspbcm
- Android (en desarrollo por usuarios)

### 2.5.1 Sistema Operativo Raspbian Wheezy

El Raspbian Wheezy es considerado el sistema operativo estándar del Raspberry Pi, tenga en cuenta que no solo es el único disponible en la red, pero sí el más usado por los usuarios que no poseen mucha experiencia utilizando Linux.

Este sistema operativo ocupa poco espacio en la memoria de la SD (2GB) debido a que apenas vienen instalados un mínimo de aplicaciones para el usuario. En otras palabras si el usuario necesita de algún programa extra u otra aplicación deberá instalarla manualmente usando líneas de comando en el LX Terminal del escritorio del Raspberry Pi.

Para aprender más acerca de la correcta instalación del sistema operativo, recomiendo visitar la siguiente página web <http://www.raspberrypi.org/downloads> la cual le ofrece de manera gratuita todas las herramientas necesarias para una instalación exitosa del Raspbian Wheezy.

## **2.6 RANGO DE TEMPERATURA**

- LAN9512 de 0°C a 70°C
- AP de -40°C a 85°C

## **2.7 DIMENSIONES**

- 85.60mm x 53.98mm x 17mm



## 2.8 ARDUINO SHIELD



Figura 2.8.Arduinoshield [2]

El arduinoshield nos ayudara hacer puente de conexión de pantallas, para permitir el uso de cualquiera de los escudos, placas y módulos diseñados para el Arduino en Raspberry Pi. Incluye también la posibilidad de conectar sensores analógicos y digitales, utilizando el mismo pin out de Arduino pero con el poder y la capacidad de frambuesa.

Con el fin de completar la compatibilidad hemos creado la biblioteca **arduPi** que permite usar la frambuesa con el mismo código utilizado en

Arduino poder ejecutarlo desde nuestra raspberry pi, y así hacer funcionar cualquier módulo que sea compatible con arduino, en este caso usaremos el módulo GPRS/GSM Quadband (SIM900).

### **2.8.1 ARDUINOSHIELD Y SUS COMPONENTES**

- 8 pines digitales.
- Zócalo para módulos inalámbricos.
- RX / TX pins.
- pines I2C (SDA, SCL).
- Pines SPI (SCK, MISO, MOSI, CS). Se puede utilizar también como GPIO.
- 8 canales de convertidor de analógico a digital.
- Conector para que la fuente de alimentación sea externa.



Figura 2.8.1.Arduinoshield y sus conectores [3]

En la figura 2.8.2 observamos el conector del arduinoshield el que logra hacer la interfaz con nuestra raspberry pi, y podemos comenzar a programar los códigos de arduino en nuestra raspberry pi con la librería ardupi la cual nos ayudara que se ejecute sin ningún problema.

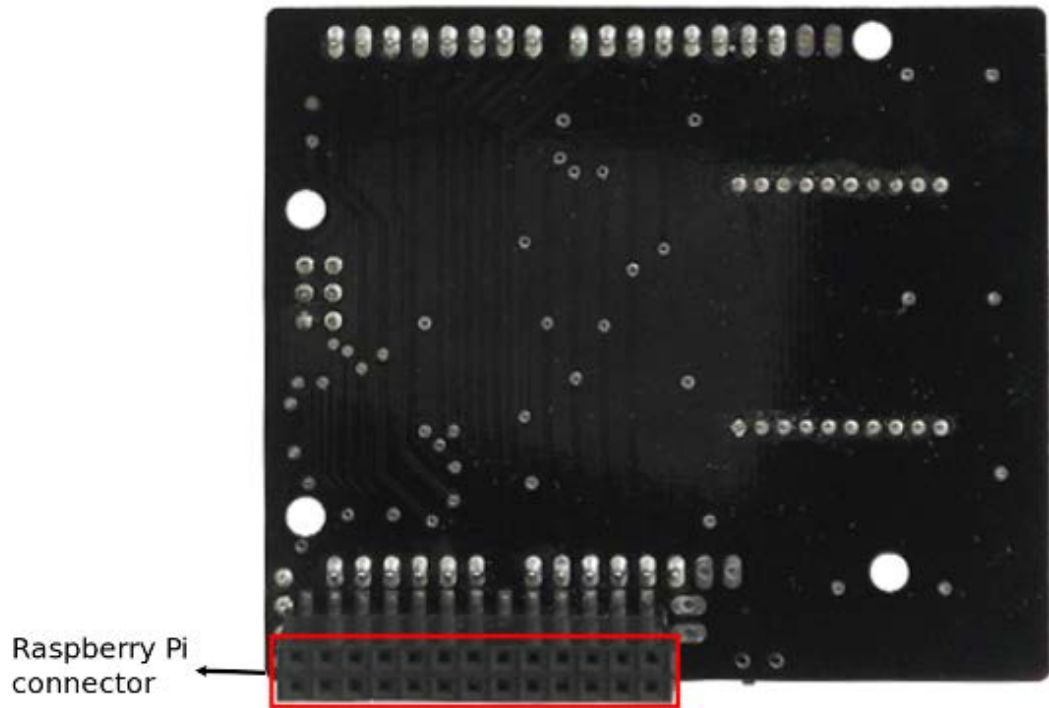


Figura 2.8.2. Conector del arduinoshield a raspberry pi [4]

## 2.9 HERRAMIENTA DE PROGRAMACIÓN

### 2.9.1 SISTEMA OPERATIVO LINUX

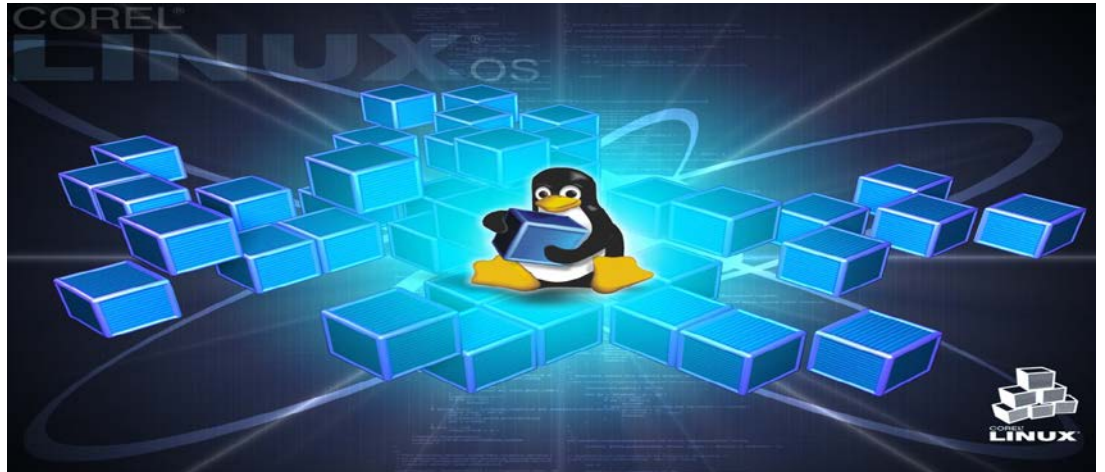


Figura 2.9.1.Sistema operativo Linux [5]

De todos los sistemas Unix, Linux es una plataforma servidora excelente, un buen sistema de escritorio y el centro en torno al cual gira una gran parte de la innovación del mundo informático actual. Linux es probablemente el que más ámbitos abarca de todos los sistemas operativos, desde sistemas pequeños como un teléfono móvil hasta clústeres de computadores más grandes que un edificio. Está presente en los campos de las telecomunicaciones, sistemas embebidos, satélites,

equipamiento médico, sistemas militares y gráficos por computador e informática de escritorio.

### **2.9.2 SISTEMA OPERATIVO DEBIAN**

Es uno de los Sistemas Operativos más estables En la actualidad. Casi no existen los malware o virus para este Sistema Operativo ,no es necesario piratear, ni crackear nada ya que el software el gratuito, una de las grandes ventajas de DEBIAN, es que posee miles de paquetes pre-compilados estable.

DEBIAN es un sistema operativo (S.O) de libre distribución, permite a todos los usuarios acceder al mismo tiempo a través de terminales y distribuye los recursos disponibles entre todos, puede corree e la mayoría de plataforma del mercado como procesadores de la gama INTEL y AMD, MOTOROLA, etc.

Los problemas de seguridad se solucionan rápidamente con parches de seguridad que se actualizan en el internet.

## 2.9.3 PROGRAMACION EN PYTHON

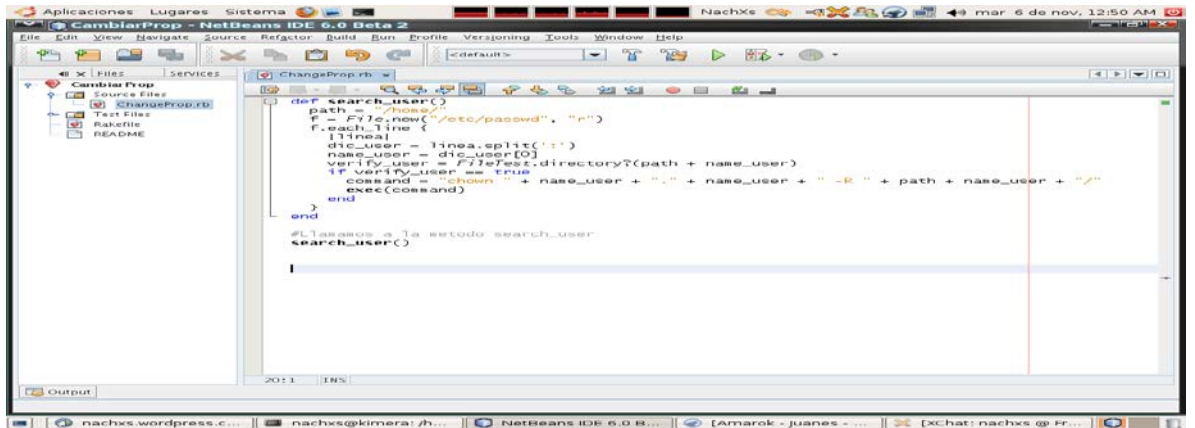


Figura 2.9.3. Programación en Python [6]

La sintaxis de Python es extremadamente sencilla. Ilustremos esa sencillez con un ejemplo: el tradicional ((Hola, mundo.)) con el que se presentan muchos lenguajes de programación. Un lenguaje como C obliga a incorporar la cabecera de una biblioteca estándar, a definir una función principal que devuelve un valor nulo (obligatorio en C99) y a codificar el salto de línea de una forma críptica:

```
#include <stdio.h>

int main (void) {
    Printf ("Hola, mundo.\n");
    Return 0;
}
```

## 2.10 MÓDULO GPRS/GSM QUADBAND (SIM900)



Figura 2.10. Módulo GPRS/GSM Quadband (SIM900) [7]

El módulo gsm, SIM900 (Fig.5) es un quad-band GSM / GPRS motor que funciona en las frecuencias de GSM 850MHz, 900MHz EGSM, DCS 1800MHz y 1900MHz PCS. SIM900 funciones GPRS multi-slot clase 10 / clase 8 (opcional) y es compatible con los esquemas de codificación GPRS CS-1, CS-2, CS-3 y CS-4. Con una configuración pequeña de 24 mm x 24 mm x 3 mm, SIM900 puede satisfacer casi la totalidad de los requisitos de espacio en sus aplicaciones, como M2M, teléfonos inteligentes, PDA y otros dispositivos móviles. La interfaz física para la aplicación móvil es una plataforma de 68-pin SMT, que proporciona todas las interfaces de hardware entre el módulo y placas de los clientes. • El teclado y la interfaz SPI pantalla le dan la flexibilidad para desarrollar aplicaciones personalizadas. Puerto serie y el puerto de depuración puede ayudarle fácilmente a desarrollar sus aplicaciones. Un canal de



audio incluye una entrada de micrófono y una salida de altavoz. SIM900 está diseñada con el poder ahorro de la técnica a fin de que el consumo de corriente es bajo como 1,5 mA en el modo SLEEP. SIM900 está integrado con el protocolo TCP / IP; extendido TCP / IP comandos AT son desarrollados para los clientes a utilizar el protocolo TCP / IP con facilidad, lo cual es muy útil para aquellas aplicaciones de transferencia de datos.

#### **2.10.1 CARACTERISTICA**

- Funciona Quad-Band 850/ 900/ 1800/ 1900 MHz
- Control mediante comandos AT (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands)
- Bajo consumo de potencia: 1.5mA(modos dormido)
- Temperatura de operación: -40°C a +85 °C
- Tamaño de placa: 6.4 x 5.1 x 1.2cm
- Nivel de voltaje de operación: Digital 5V
- Voltaje de alimentación externa: 9V, 12V, 24V

### 2.10.2 FUENTE DE ALIMENTACION

La fuente de alimentación de SIM900 es de una sola fuente de voltaje de 3.4V. En algún caso, el rizado en una ráfaga de transmisión puede causar caídas de tensión cuando el consumo de corriente aumenta a picos típicos de 2A. Así que la fuente de alimentación debe ser capaz de proporcionar suficiente corriente hasta 2A.

Para la entrada de VBAT, un condensador de derivación local se recomienda. Un condensador (alrededor de 100 mF, bajo ESR) se recomienda. Multi-capas de cerámica chip (MLCC) los condensadores puede proporcionar la mejor combinación de tamaño bajo ESR y pequeño, pero puede no ser rentable. Una opción de menor costo puede ser un condensador de Tántalo 100 mF (ESR bajo) con un pequeño (0.1 $\mu$ F a 1 $\mu$ F) de cerámica en paralelo, que se ilustra como la siguiente figura. Los condensadores deben estar colocados tan cerca como sea posible a los SIM900 pins VBAT. La figura 2.10.2 es el circuito recomendado.

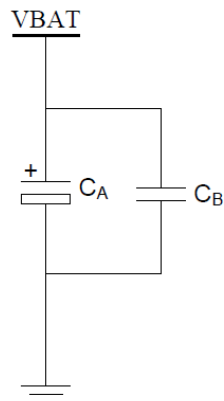


Figura 2.10.2.Circuitos de alimentación [8]

El diseño del circuito de la fuente de alimentación depende en gran medida de la fuente de alimentación donde se drena esta potencia. La figura siguiente es el diseño de referencia de 5 V de alimentación de entrada de fuente de alimentación. La salida diseñada para la fuente de alimentación es 4,1 V, por lo tanto un regulador lineal se puede utilizar. Si hay una gran diferencia entre la fuente de entrada y la salida deseada (VBAT), un convertidor de fuente de alimentación de conmutación será preferible debido a su mayor eficiencia sobre todo con el pico de 2A de corriente en modo de ráfaga del módulo.

El único 3.6V Li-Ion Tipo de batería de la célula puede ser conectado a la fuente de alimentación de la VBAT SIM900 directamente. Sin embargo, los tipos de baterías o Ni\_CdNi\_MH debe usarse con cuidado, ya que su

voltaje máximo puede elevarse por encima de lo normal y pueda dañar el módulo y dañarlo.

### 2.10.3 ESQUEMATICO DEL GSM

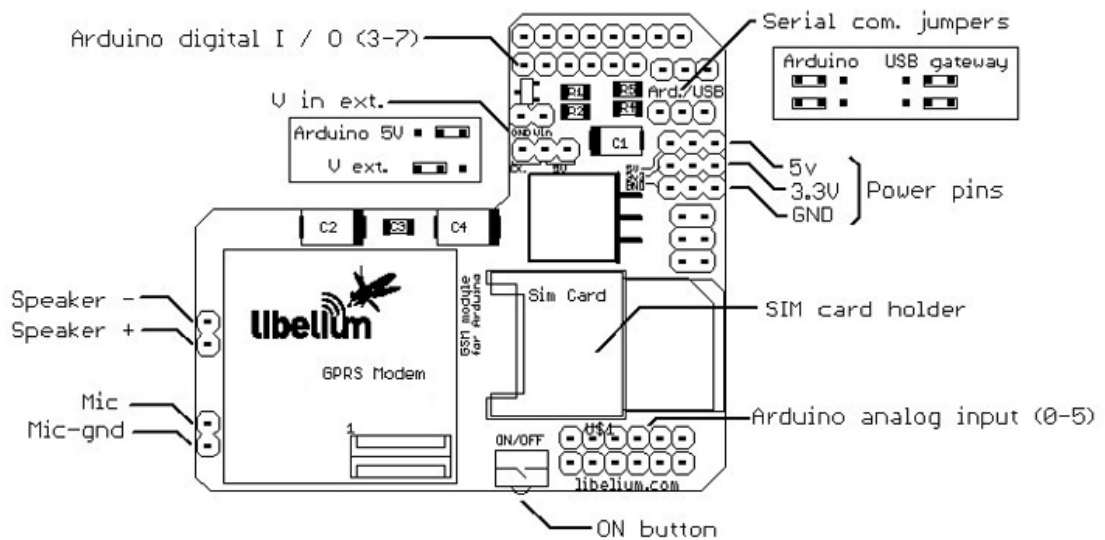


Figura 2.10.3. Diagrama del módulo GPRS/GSM Quadband (SIM900) [9]

Cuestiones importantes:

Utilice letras mayúsculas para los comandos AT. Send CR (retorno de carro) y LF (avance de línea) después de que el comando AT. Coloque los puentes de comunicación en serie en la posición correcta, utilice una

fuentes de alimentación externa y colocar los puentes de alimentación en la posición correcta. Si el escudo se alimenta de la Raspberry Pi, el puente de alimentación debe estar en posición Arduino 5V. Si el escudo se alimenta desde la entrada Vin (en la pantalla), el puente de alimentación debe estar en la posición Vext.

Lo primero que vamos a hacer con el módulo es comprobar los básicos comandos AT. En este caso, los puentes de comunicación en serie que se encuentra en posición de puerta de enlace USB.

A continuación, introducir la tarjeta SIM en el escudo GPRS / GSM.

Finalmente abra en la Raspberry terminal de un puerto serie para comunicarse con el escudo GPRS / GSM (sólo puede ejecutarse en un terminal). Este paso se puede realizar directamente desde la Raspberry pi con un teclado, ratón y pantalla, o bien a través de ssh exportador X (por ejemplo: ssh-X pi @ <dirección dirección de raspberry>)

Asegúrese de que está enviando CR (retorno de carro) y LF (Line Feed).

Ajuste la velocidad en baudios a 115200 bps y abra el puerto serie, a continuación, pulse el botón de encendido durante dos segundos.

Entonces si escribe AT obtendrá OK, esto significa que la comunicación con el módulo está funcionando bien. Ahora, con el módulo de trabajo se

puede comprobar algunos comandos AT para controlar el módulo, los comandos básicos son:

Comandos importantes de tipo en letras mayúsculas y con CR (retorno de carro) y LF (avance de línea).

Comando Descripción Respuesta

AT Aceptar Si obtiene OK, la comunicación con el módulo está funcionando

AT + CPIN = "\*\*\*\*" OK Si la tarjeta SIM está bloqueada con PIN (\*\*\*\* es el número pin)

AT + COPS? información del operador

### **Nota**

El ajuste de fábrica es baudrate auto-bauding por defecto. Velocidad de transmisión se puede fijar usando el comando AT + IPR = velocidad de transmisión. Baudrates permitidos: 0 (Auto-bauding), 1200, 2400, 4800, 9600, 19200, 38400, 57600 y 115200.

## **CAPÍTULO 3**

### **DESARROLLO DEL PROYECTO**

#### **3.1 RESUMEN**

En este capítulo se mostrará el desarrollo de nuestro proyecto, el procedimiento que vamos a realizar para poder entender el funcionamiento de cada parte de nuestro proyecto y luego mostrar cómo han sido combinadas para alcanzar nuestro objetivo propuesto, para esto empezaremos a programar el Raspberry pi, con sus diferentes puertos de entrada y salida.

Descargaremos programas y librerías que serán útiles en nuestro proyecto, comenzaremos descargando la librería (ardupi) en nuestra raspberry pi para poder hacer posible la ejecución de los código fuente, y habilitaremos el puerto uart.

1. Haga una copia de seguridad del archivo / boot / cmdline.txt. sudo cp / boot / cmdline.txt / boot / cmdline\_backup.txt

2. Editar archivo / boot / cmdline.txt: sudo vi / boot / cmdline.txt Este archivo contiene: dwc\_otg.lpm\_enable = 0 console = ttyAMA0, 115.200 kgdboc = ttyAMA0, 115200 console = tty1 \$ Retire los parámetros que hacen referencia al puerto serie UART ( ttyAMA0): dwc\_otg.lpm\_enable = 0 console = tty1 \$ .

3. Editamos la siguiente línea en / etc / inittab: T0: 23: respawn :/sbin / getty-L ttyAMA0 115200 vt100

4. Reinicie la raspberry pi sudo reboot

Luego descargaremos el programa cutecom que nos ayudara saber si el módulo GPRS/GSM Quadband (SIM900) es detectado en los puertos de la raspberrypi.

En los ejercicios siguientes, probaremos las ejecuciones con comandos que les indicaremos paso a paso en el capítulo siguiente.



## 3.2 EJERCICIO 1

### INTERFAZ DE RASPBERRY PI CON EL ARDUINO SHIELD

En este ejercicio 1 comprobaremos la compatibilidad de la Raspberry pi, con el arduinoshield en cual consiste en prender y apagar un led con una configuración básica, y con su respectiva librería (ardupi) gracias a esta librería podemos programar y usar el arduinoshield y poder acoplar módulos, en este caso nuestro proyecto usaremos el módulo GPRS/GSM Quadband (SIM900).

#### 3.2.1 DIAGRAMA DE BLOQUE

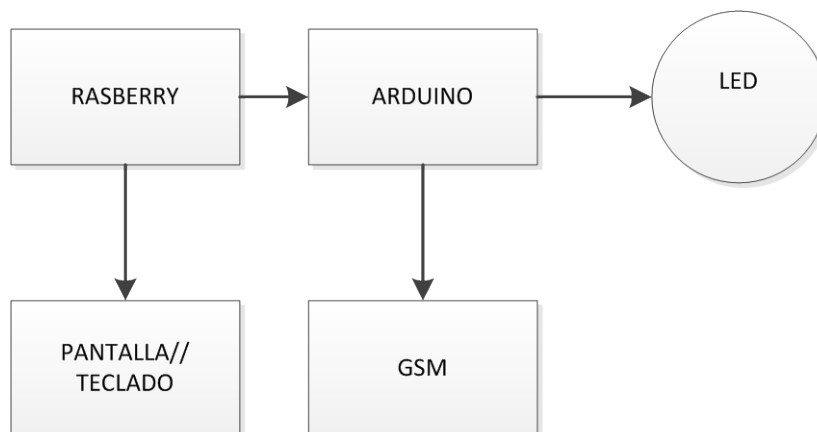


Figura 3.2.1. Diagrama de bloque ejercicio 1

### 3.2.2 CÓDIGO FUENTE

```
//Include ArduPi library
#include "arduPi.h"

//Needed for Serial communication
SerialPi Serial;

//Needed for accesing GPIO (pinMode, digitalWrite, digitalRead, I2C
functions)
WirePi Wire;

//Needed for SPI
SPIPi SPI;

int main (){
  setup();
  while(1){
    loop();
  }
  return (0);
}

void setup(){
  pinMode(2,OUTPUT);
}

void loop(){
  digitalWrite(2,HIGH);
```

```
delay(1000);  
digitalWrite(2,HIGH);  
delay(1000);  
digitalWrite(2,LOW);  
delay(1000);  
} [10]
```

### **3.2.3 CONCLUSIÓN**

En este ejercicio podemos concluir la compatibilidad del Raspberry pi con el arduino shield, gracias a esta importante librería (ardupi) que hace posible la compatibilidad de cualquier programación en arduino hacerla ejecutar por medio de la interfaz de Raspberry pi y arduinoshield.

## **3.3 EJERCICIO 2**

### **CONTROLAR 4 LED CON LOS GPIO DE LA RASPBERRY PI**

En el ejercicio 2 controlaremos 4 LED en una secuencia que nos permitirá contar en binario del cero al nueve y además rutear esa salida hacia un circuito integrado 7447 para poder desplegar los dígitos en un display de 7 segmento

### 3.3.1 DIAGRAMA DE BLOQUE

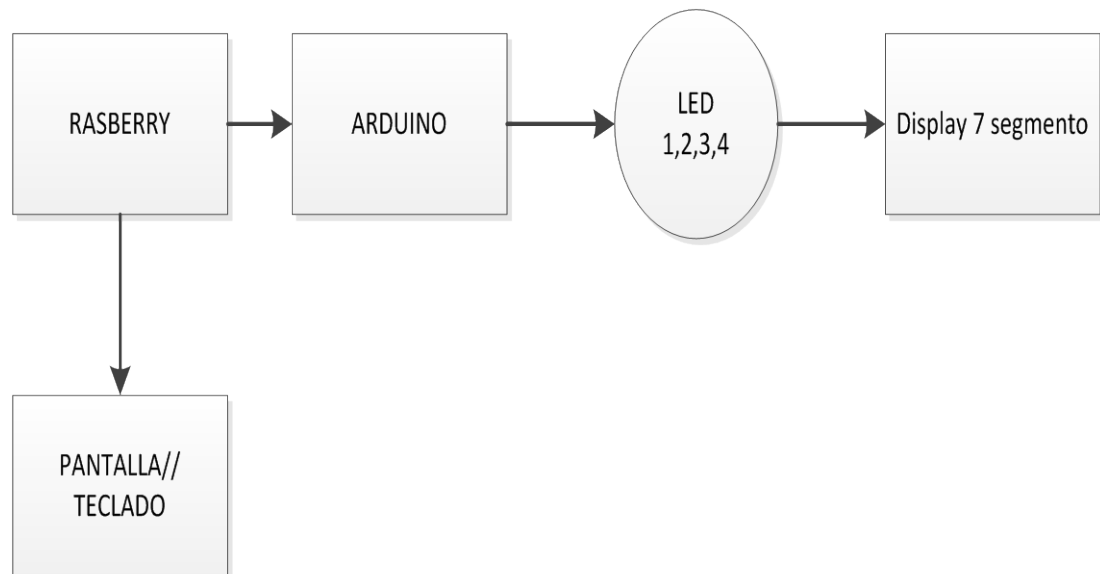


Figura 3.3.1. Diagrama de bloque ejercicio 2

### 3.3.2 CÓDIGO FUENTE

```
#!/usr/bin/env python
# Import required libraries
import time
import math
import RPi.GPIO as GPIO

# Set up the GPIO mode
GPIO.setmode(GPIO.BCM)
```

```
#Initialize a list with the GPIO pin numbers we are going to use
gpiopins = [14,15,18,23]

# Set up pins listed as outputs
for p in gpiopins :
    print "Setup GPIO" + str(p) + " for OUTPUT"
    GPIO.setup(p, GPIO.OUT)
    GPIO.setup(p, GPIO.LOW)
print "Setup completed."

#Initialize a list with the binary values for the first 10 decimal values
binary = [
[0,0,0,0],[0,0,0,1],[0,0,1,0],[0,0,1,1],[0,1,0,0],[0,1,0,1],[0,1,1,0],[0,1,1,1],[1,0
,0,0],[1,0,0,1] ]

print "Start loop"
while True :
    digit = 0
    while digit < 10 :
        print "Decimal: " + str(digit) + " Binary: " + str(binary[digit])
        for x in range(0,4) :
            GPIO.output(gpiopins[x], binary[digit][x])
```

```
print "pin = " + str(gpiopins[x]) + " valor = " + str(binary[digit][x])  
  
time.sleep(1)  
  
digit = digit + 1
```

### 3.3.3 CONCLUSIÓN

En este ejercicio podemos concluir que podemos desarrollar sin número de aplicaciones usando el puerto gpio de nuestra raspberry pi como este ejercicio que podemos usar los puertos en conjuntos o por si solos, cada puerto con su función correspondiente que serán muy útiles para finalizar nuestro proyecto.

### 3.4 EJERCICIO 3

#### PROBANDO EL MÓDULO GPRS/GSM QUADBAND (SIM900) CON EL PROGRAMA CUTECOM.

En el ejercicio 2 usaremos un programa (cutecom) para detectar el módulo mediante nuestra Raspberry pi, y con comandos podemos verificar si el módulo GPRS/GSM Quadband (SIM900) está funcionando correctamente es la combinación de dos letras que es llamado el comando AT.

##### 3.4.1 DIAGRAMA DE BLOQUE



Figura 3.4.1. Diagrama de bloque ejercicio 3

### 3.4.2 CONCLUSIÓN

En este ejercicio 2 comprobaremos si nuestro módulo GPRS/GSM Quadband (SIM900) está funcionando correctamente y poder lograr manipularlo con una programación indicada.

### 3.5 EJERCICIO 4

#### INTERFAZ ENTRE EL MÓDULO GPRS/GSM QUADBAND (SIM900) ARDUINO SHIELD Y RASPBERRY PI

En este ejercicio comprobaremos la compatibilidad del módulo GPRS/GSM sim900 con el ARDUINO SHIELD y la RASPBERRY PI mediante una programación en la cual consiste que el módulo GPRS/GSM sim900 mande un mensaje de texto a un celular móvil.

#### 3.5.1 DIAGRAMA DE BLOQUE



Figura 3.5.1. Diagrama de bloque ejercicio 4



### 3.5.2 CÓDIGO FUENTE

```

//Include ArduPi library

#include "arduPi.h"

Void switchModule ();

Int led = 9;

Int onModulePin = 2;    // the pin to switch on the module (without press
on button)

Int timesToSend = 1;    // Numbers of SMS to send

Int count = 0;

Char phone_number []="*****";    // ***** is the number to call

VoidswitchModule (){

Digital Write (onModulePin, HIGH);

Delay (2000);

digital Write(onModulePin, LOW);

}

Void setup (){

Serial. Begin(115200);    // UART baud rate

Delay (2000);

PinMode (led, OUTPUT);

PinMode (onModulePin, OUTPUT);

SwitchModule ();    // switches the module ON

For (int i=0;i < 5;i++){

```

```
Delay (5000);  
    }  
Serial.println ("AT+CMGF=1");    // sets the SMS mode to text  
Delay (100);  
}  
Voidloop (){  
While (count < timesToSend){  
Delay (1500);  
Serial. Print ("AT+CMGS=\"");    // send the SMS number  
Serial. Print (phone_number);  
    Serial.println("\");  
While (Serial. Read ()!='>');  
Serial.print ("Hola caracola...");    // the SMS body  
Delay (500);  
Serial.write (0x1A);    //sends ++  
    Serial.write(0x0D);  
Serial.write (0x0A);  
Delay (5000);  
    count++;  
    }  
}  
int main (){
```

```

Setup ();
While (1){
    loop();
}
return (0);
} [11]

```

### 3.5.3 CONCLUSIÓN

En este ejercicio podemos concluir la compatibilidad del módulo GPRS/GSM sim900 con el ARDUINO SHIELD y la RASPBERRY PI, con nuestra programación comprobamos que el módulo sim900 funciona perfectamente y está listo para continuar la otra etapa del proyecto.

### 3.6 PROYECTO TERMINADO

Todos los ejercicios mencionados anteriormente ayudaron a la culminación de nuestro proyecto que consiste en la activación de una alarma y enviar un mensaje de advertencia.

#### 3.6.1 DIAGRAMA DE BLOQUES

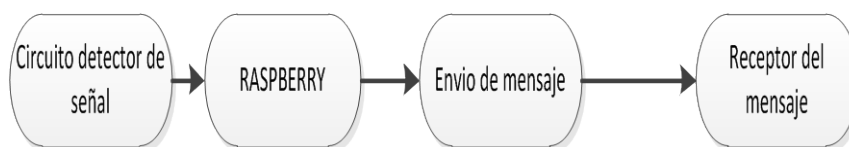


Figura 3.6.1. Diagrama de bloques del proyecto terminado

### 3.6.2 DIAGRAMA DE FLUJO

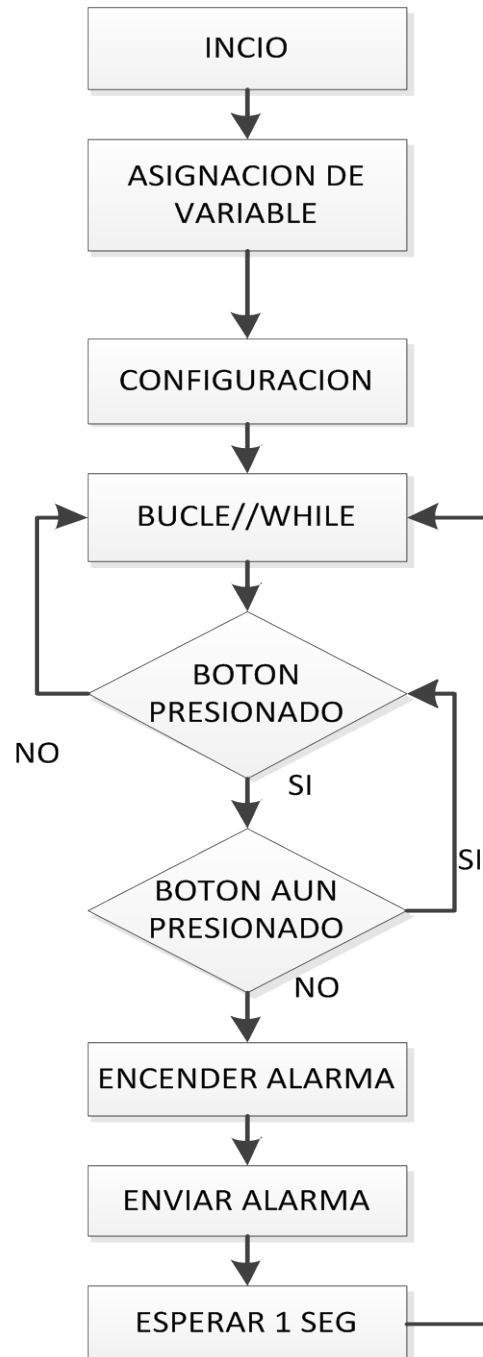


Figura 3.6.2(A).Diagrama de flujo del proyecto terminado

## ENVIAR MENSAJE

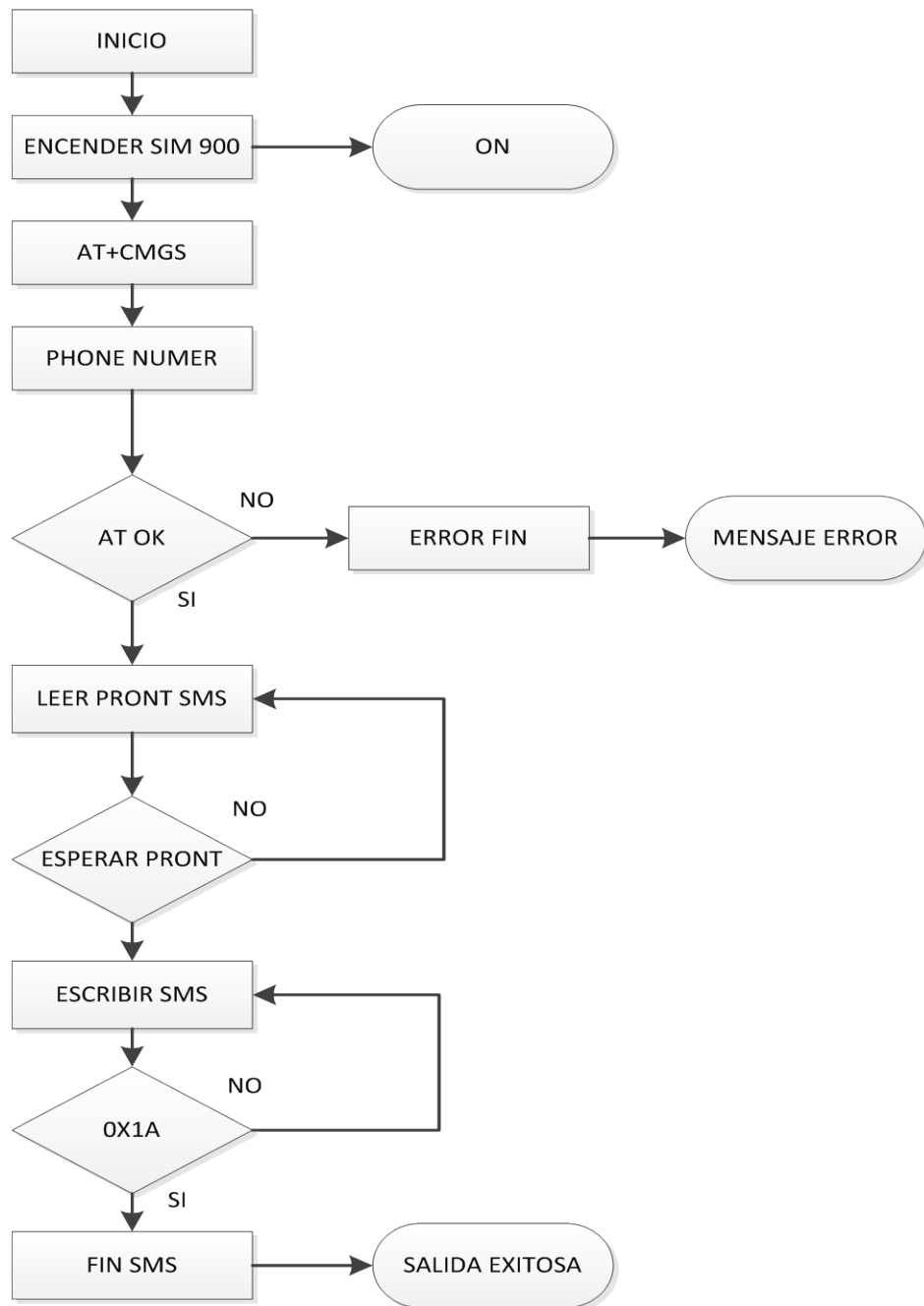


Figura 3.6.2 (B).Diagrama de flujo del proyecto terminado

### 3.6.3 CÓDIGO FUENTE

```
//Include ArduPi library

#include "arduPi.h"

#include <wiringSerial.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

void encenderAlarma();

void enviarSMS();

void enviarSMS2();

void switchModule();

SerialPi Serial;

WirePi Wire;

int PIN_ALARMA = 8;

int PIN_BOTON = 9;

int onModulePin = 2; // the pin to switch on the module (without
press on button)

char phone_number [] = "0990518541";

int main()

{

Setup°();
```

```
    while(1){  
loop ();  
    }  
return (0);  
}  
  
void setup()  
{  
pinMode (PIN_ALARMA, OUTPUT);  
    pinMode(PIN_BOTON, INPUT);  
  
digitalWrite(PIN_ALARMA,LOW);  
    delay(2000);  
    digitalWrite(PIN_ALARMA,HIGH);  
    delay(500);  
    digitalWrite(PIN_ALARMA,LOW);  
  
Serial.begin(115200);          // UART baud rate  
    delay(2000);  
    pinMode(onModulePin, OUTPUT);  
    switchModule();           // switches the module ON  
    for (int i=0;i < 5;i++){  
        delay(2000);  
    }  
}
```

```
}

    printf("Se inicio el programa \n");
    Serial.println("AT+CMGF=1");    // sets the SMS mode to text
    delay(100);
}

void loop()
{
    if(digitalRead(PIN_BOTON) == 0){
    while(digitalRead(PIN_BOTON) == 0);
        printf("El boton fue presionado\n");
        encenderAlarma();
        enviarSMS();
    }
    delay(1000);
}

void encenderAlarma()
{
    printf("La alarman se encendio\n");
    digitalWrite(PIN_ALARMA,HIGH);
    delay(1000);
    digitalWrite(PIN_ALARMA,LOW);
}
```



```
    delay(1000);
}

void enviarSMS()
{
    printf("Se envio un sms por el modem ardipi\n");
    Serial.print("AT+CMGS=\""); // send the SMS number
    Serial.print(phone_number);
    Serial.println("\"");
    while(Serial.read()!='>');
    Serial.print("La alarma se ha activado..."); // the SMS body
    delay(500);
    Serial.write(0x1A); //sends ++
    Serial.write(0x0D);
    Serial.write(0x0A);
    delay(5000);
}

void switchModule()
{
    digitalWrite(onModulePin,HIGH);
    delay(2000);
    digitalWrite(onModulePin,LOW);
    printf("Se activó el modem ardipi\n");
```

```
}  
  
void enviarSMS2()  
{  
    printf("Se envio un sms por el modem huawei\n");  
    int fd ;  
    if ((fd = serialOpen ("/dev/ttyUSB2", 115200)) < 0){  
        //if ((fd = serialOpen ("/dev/ttyAMA0", 115200)) < 0){  
            fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno)) ;  
        }  
        SerialPrintf (fd,"AT\r");  
        SerialPrintf (fd,"AT+CMGF=1\r");  
        SerialPrintf (fd,"AT+CMGS=\"");  
        SerialPrintf (fd,phone_number);  
        SerialPrintf (fd,"\\r");  
        SerialPrintf (fd,"La alarma se ha activado...");  
        SerialPrintf (fd,"\\x1A");  
    }  
}
```

### **3.6.4 CONCLUSIÓN**

Podemos concluir que nuestra programación funciona perfectamente en nuestras interfaces que desarrollamos anteriormente el módulo GPRS/GSM Quadband (SIM900) envía el mensaje de texto sin ningún problema en un tiempo muy bueno aproximadamente 5 segundos, después de haberse activado la alarma y podemos así tomar decisiones y evitar pérdidas en el lugar que ha sido instalada, nuestro sistema de alarma.

## **CAPÍTULO 4**

### **SIMULACIONES Y PRUEBAS EXPERIMENTADAS**

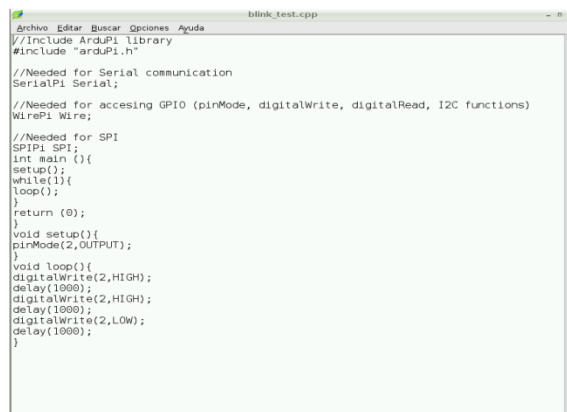
#### **4.1 RESUMEN**

Este capítulo se basa en las diferentes pruebas que se realizaron para finalizar nuestro proyecto. A continuación mostramos los resultados obtenidos en cada uno de los ejercicios propuestos en el capítulo anterior que son los pasos para poder cumplir con los objetivos de este tema de proyecto.

En cada ejercicio mostraremos imágenes con su debida explicación de cómo se realiza la misma y luego los resultados de la implementación real con una breve descripción del esquema.

## 4.2 IMÁGENES DEL EJERCICIO 1

La figura 4.2 (a) y (b) muestra la complicación del código del ejercicio 1 capítulo 3, en el terminal de nuestra Raspberry pi con el comando ( `g++ -lrt -lpthread arduPi_template.cpp arduPi.o -o test` )

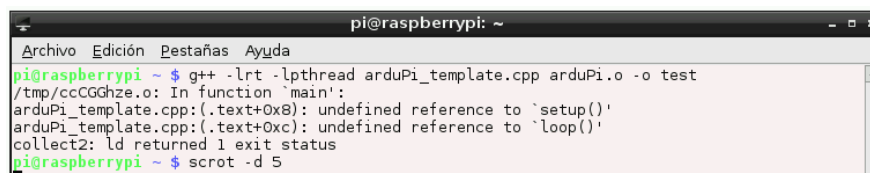


```

blink_test.cpp
Archivo Editar Buscar Opciones Ayuda
//include arduPi library
#include "arduPi.h"
//Needed for Serial communication
SerialPI Serial;
//Needed for accessing GPIO (pinMode, digitalWrite, digitalRead, I2C functions)
WirePI Wire;
//Needed for SPI
SPIPI SPI;
int main (){
  setup();
  while(1){
    loop();
  }
  return (0);
}
void setup(){
  pinMode(2,OUTPUT);
}
void loop(){
  digitalWrite(2,HIGH);
  delay(1000);
  digitalWrite(2,HIGH);
  delay(1000);
  digitalWrite(2,LOW);
  delay(1000);
}

```

Figura 4.2 (a).Código del ejercicio 1



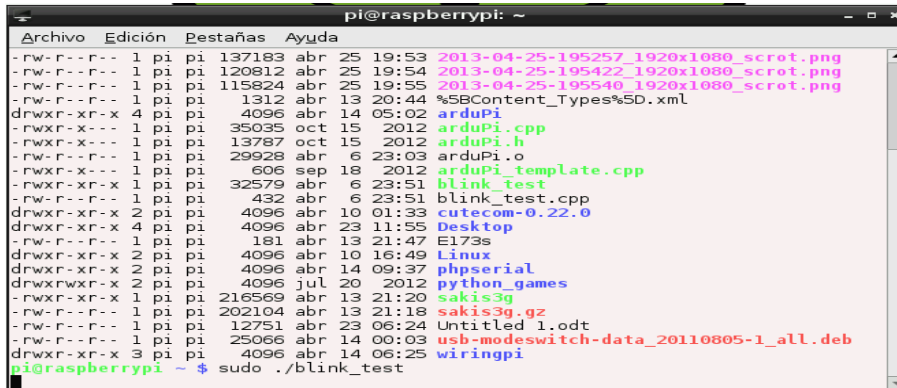
```

pi@raspberrypi: ~
Archivo Edición Pestañas Ayuda
pi@raspberrypi ~$ g++ -lrt -lpthread arduPi_template.cpp arduPi.o -o test
/tmp/ccGGhze.o: In function `main':
arduPi_template.cpp:(.text+0x8): undefined reference to `setup()'
arduPi_template.cpp:(.text+0xc): undefined reference to `loop()'
collect2: ld returned 1 exit status
pi@raspberrypi ~$ scrot -d 5

```

Figura 4.2 (b).Código compilado

La figura 4.2(c) se muestra la ejecución del código compilado de este ejercicio, se usó este comando (sudo ./blink\_test)



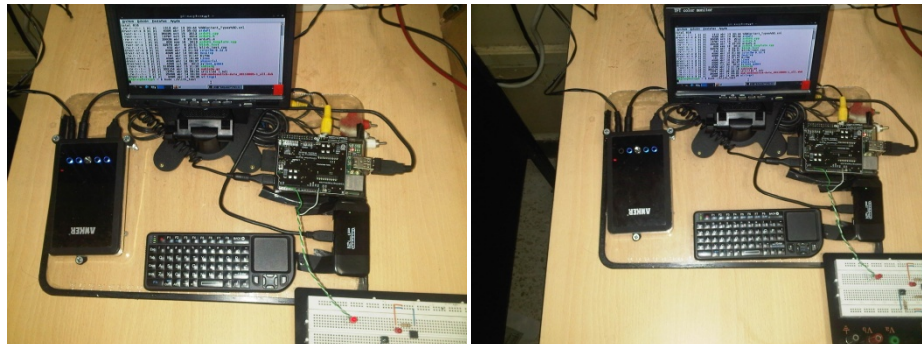
```

pi@raspberrypi: ~
Archivo Edición Pestañas Ayuda
-rw-r--r-- 1 pi pi 137183 abr 25 19:53 2013-04-25-195257_1920x1080_scrot.png
-rw-r--r-- 1 pi pi 120812 abr 25 19:54 2013-04-25-195422_1920x1080_scrot.png
-rw-r--r-- 1 pi pi 115824 abr 25 19:55 2013-04-25-195546_1920x1080_scrot.png
-rw-r--r-- 1 pi pi 1312 abr 13 20:44 %5BContent_Type%5D.xml
drwxr-xr-x 4 pi pi 4096 abr 14 05:02 arduPi
-rwxr-x-- 1 pi pi 35035 oct 15 2012 arduPi.cpp
-rwxr-x-- 1 pi pi 13787 oct 15 2012 arduPi.h
-rw-r--r-- 1 pi pi 29928 abr 6 23:03 arduPi.o
-rwxr-x-- 1 pi pi 606 sep 18 2012 arduPi_template.cpp
-rwxr-xr-x 1 pi pi 32579 abr 6 23:51 blink_test
-rw-r--r-- 1 pi pi 432 abr 6 23:51 blink_test.cpp
drwxr-xr-x 2 pi pi 4096 abr 10 01:33 cutescom-0.22.0
drwxr-xr-x 4 pi pi 4096 abr 23 11:55 Desktop
-rw-r--r-- 1 pi pi 181 abr 13 21:47 E173s
drwxr-xr-x 2 pi pi 4096 abr 10 16:49 Linux
drwxr-xr-x 2 pi pi 4096 abr 14 09:37 phpserial
drwxrwxr-x 2 pi pi 4096 jul 20 2012 python_games
-rwxr-xr-x 1 pi pi 216569 abr 13 21:20 sakis3g
-rw-r--r-- 1 pi pi 202104 abr 13 21:18 sakis3g.gz
-rw-r--r-- 1 pi pi 12751 abr 23 06:24 Untitled 1.odt
-rw-r--r-- 1 pi pi 25066 abr 14 09:03 usb-modeswitch-data_20110805-1_all.deb
drwxr-xr-x 3 pi pi 4096 abr 14 06:25 wiringpi
pi@raspberrypi ~ $ sudo ./blink_test

```

Figura 4.2 (c).Ejecutando programa

La figura 4.2 (d) se muestra el programa ejecutado y lo podemos visualizar físicamente el desarrollo de nuestro programa.



Led (encendido)

Led (apagado)

Figura 4.2 (d).Imágenes del ejercicio terminado

### 4.3 SIMULACIÓN EJERCICIO 2

La figura 4.3 muestra la compilación del código del ejercicio 2 capítulo 3, en el terminal de nuestra Raspberrypi.

```

#!/usr/bin/env python
# Import required libraries
import time
import math
import RPi.GPIO as GPIO

# Set up the GPIO mode
GPIO.setmode(GPIO.BCM)

# Initialize a list with the GPIO pin numbers we are going to use
gpioPins = [14, 15, 18, 23]

# Set up pins listed as outputs
for p in gpioPins:
    print "Setup GPIO" + str(p) + " for OUTPUT"
    GPIO.setup(p, GPIO.OUT)
    print "Setup completed."

# Initialize a list with the binary values for the first 18 decimal values
binary = [ (0,0,0,0), (0,0,0,1), (0,0,1,0), (0,0,1,1), (0,1,0,0), (0,1,0,1), (0,1,1,0), (0,1,1,1), (1,0,0,0), (1,0,0,1) ]

print "Start loop"
while True:
    digit = 0
    while digit < 18:
        print "Decimal: " + str(digit) + " binary: " + str(binary[digit])
        for x in range(0,4):
            GPIO.output(gpioPins[x], binary[digit][x])
            print "pin " + str(gpioPins[x]) + " valor = " + str(binary[digit][x])
            time.sleep(1)
        digit = digit + 1

```

Figura 4.3(a).Código del ejercicio2

En la figura 4.3 (b) observamos el diseño del ejercicio 2 de la raspberry pi de los puertos gpio conectado con el protoboard con nuestro circuito.

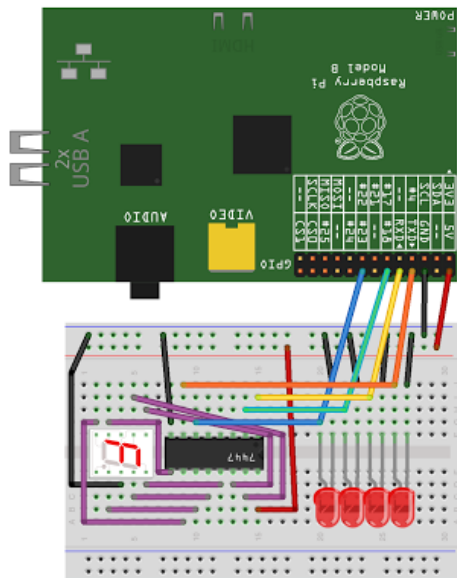


Figura 4.3 (b).Diseño en protoboard.

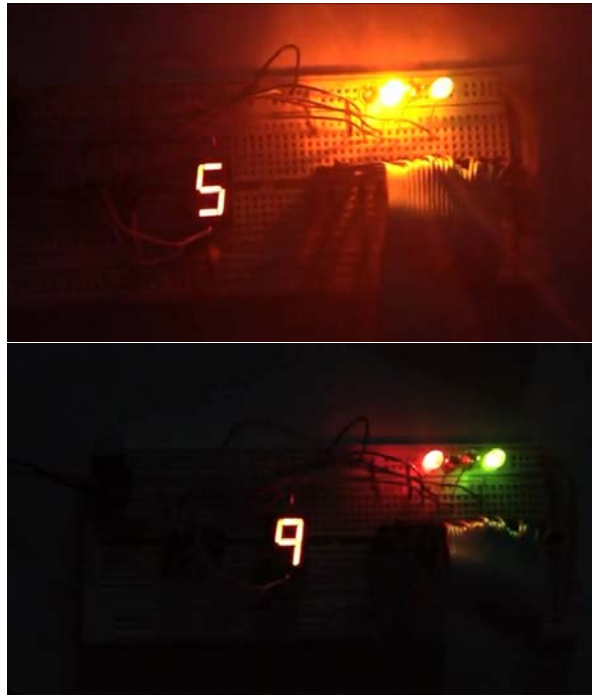


Figura 4.3 (c).Ejercicio Ejecutado

#### 4.4 SIMULACIÓN EJERCICIO 3

En la figura 4.3 observaremos como el programa cutecom detecta el módulo GPRS/GSM Quadband (SIM900) con el comando AT.

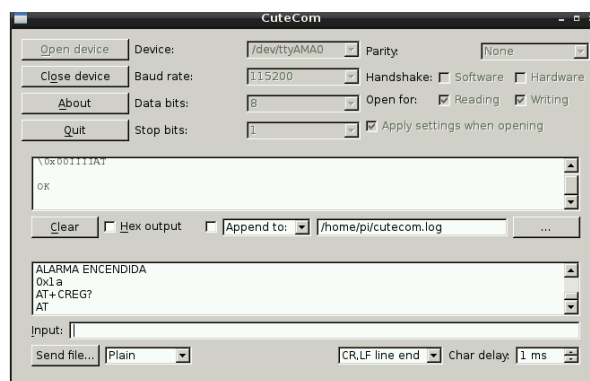


Figura 4.4(a).Detectando el módulo sim900 con comando AT=ok



En la figura 4.4 (b) observaremos como mandar un mensaje de texto mediante el programa cutecom, con el módulo GPRS/GSM Quadband (SIM900) con los comandos siguientes.

- AT+CMGF=1

La función de este comando es dar la orden de que se va configurar el módulo sim900 para mandar un mensaje de texto.

-AT+CMGS=" \_\_\_\_\_"

La función de este comando nos servirá para configurar el número de un celular móvil que queremos que llegue el mensaje de texto.

->MENSAJE DE TEXTO

Después de haber agregado el número nos aparecerá un > que nos indica escribir cualquier mensaje que se quiera enviar en este caso "ALARMA ENCENDIDA"

-0x1a

La función de este comando dará la orden de que se envíe el mensaje, el cual será recibido en el número programado anteriormente.

Nota: al ejecutar este comando hay que cambiar la opción de CR, LF line end por HEX.

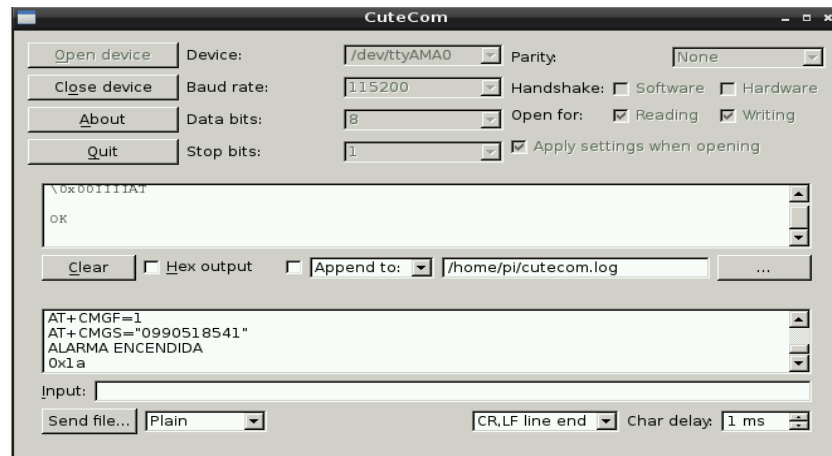


Figura 4.4 (b). Verificación de transmisión del módulo sim900

En la fig. 4.4 (c) se muestra el mensaje enviado por medio del programa cutecom a un celular móvil.

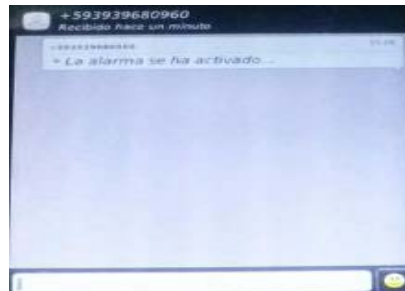


Figura 4.4 (c). Muestra el mensaje recibido por medio del programa cutecom [18]

## 4.5 SIMULACIÓN EJERCICIO 4

La figura 4.5(a) y (b) muestra la simulación Interfaz entre el módulo GPRS/GSM Quadband (SIM900) arduino shield y Raspberry Pi con el código compilado y ejecutado en la terminal de la raspberry pi.



```

#include <Arduino.h>
#include <string.h>
#include <string>
#include <stringstream>
#include <string.h>
#include <string.h>
#include <string.h>

void encenderAlarma();
void enviarSMS();
void enviarSMS2();
void switchModule();

SerialPI Serial;
WirePI Wire;
int PIN_ALARMA = 8;
int PIN_BOTON = 9;
int orModulePin = 2; // the pin to switch on the module (without press on button)
char phone_number[] = "8998518541";

int main()
{
  setup();
  while(1){
    loop();
  }
  return(0);
}

void setup()
{
  pinMode(PIN_ALARMA, OUTPUT);
  pinMode(PIN_BOTON, INPUT);
  digitalWrite(PIN_ALARMA, LOW);
  delay(2000);
  digitalWrite(PIN_ALARMA, HIGH);
  delay(500);
  digitalWrite(PIN_ALARMA, LOW);

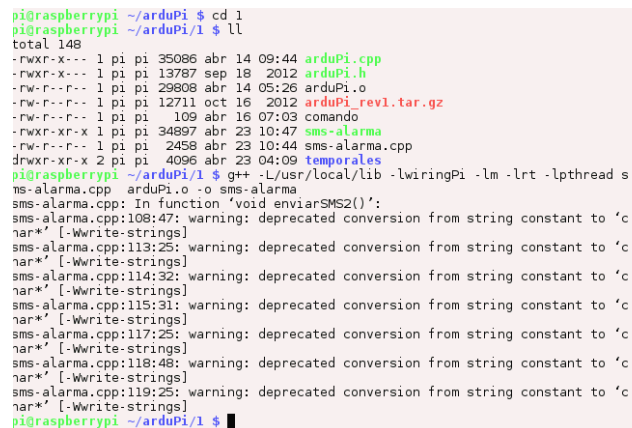
  Serial.begin(115200); // UART baud rate
  delay(2000);
  pinMode(orModulePin, OUTPUT); // switches the module ON
  switchModule();

  for (int i=0; i < 5; i++){
    delay(2000);
  }

  printf("Se inicio el programa \n");
  Serial.println("AT+CMGF=1"); // sets the SMS mode to text
  delay(100);
}

```

Figura 4.5(a).Código



```

pi@raspberrypi ~/arduPi $ cd 1
pi@raspberrypi ~/arduPi/1 $ ll
total 148
-rwxr-x--- 1 pi pi 35086 abr 14 09:44 arduPi.cpp
-rwxr-x--- 1 pi pi 13787 sep 18 2012 arduPi.h
-rw-r--r-- 1 pi pi 29808 abr 14 05:26 arduPi.o
-rw-r--r-- 1 pi pi 12711 oct 16 2012 arduPi_rev1.tar.gz
-rw-r--r-- 1 pi pi 109 abr 16 07:03 comando
-rwxr-xr-x 1 pi pi 34897 abr 23 10:47 sms-alarma
-rw-r--r-- 1 pi pi 2458 abr 23 10:44 sms-alarma.cpp
drwxr-xr-x 2 pi pi 4096 abr 23 04:09 temporales
pi@raspberrypi ~/arduPi/1 $ g++ -L/usr/local/lib -lwiringPi -lm -lrt -pthread sms-alarma.cpp arduPi.o -o sms-alarma
sms-alarma.cpp: In function 'void enviarSMS2()':
sms-alarma.cpp:108:47: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
sms-alarma.cpp:113:25: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
sms-alarma.cpp:114:32: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
sms-alarma.cpp:115:31: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
sms-alarma.cpp:117:25: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
sms-alarma.cpp:118:48: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
sms-alarma.cpp:119:25: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
pi@raspberrypi ~/arduPi/1 $

```

Figura 4.5 (b).Compilación y ejecución del programa

## 4.6 SIMULACIÓN PROYECTO

La figura 4.6 (a) se muestra la compilación y la ejecución de nuestro código final para el envío del mensaje de texto por la activación de una alarma de seguridad por medio de la interfaz del módulo GPRS/GSM sim900 con el ARDUINO SHIELD y la RASPBERRY PI.

```

pi@raspberrypi ~/arduino/1 $ g++ -L/usr/local/lib -lwiringPi -lm -lrt -lpthread sms-alarma.cpp arduino.o -o sms-alarma
sms-alarma.cpp: In function 'void enviarSMS2()':
sms-alarma.cpp:108:47: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
sms-alarma.cpp:113:25: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
sms-alarma.cpp:114:32: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
sms-alarma.cpp:115:31: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
sms-alarma.cpp:117:25: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
sms-alarma.cpp:118:48: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
sms-alarma.cpp:119:25: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
pi@raspberrypi ~/arduino/1 $ sudo ./sms-alarma

```

Figura 4.6 (a) Compilación y ejecución del código final

La figura 4.6 (b) y (c) muestra el funcionamiento de todo el proyecto, ejecutamos el programa y nos mostrara un mensaje en el terminal de la raspberry pi, que se activó el módulo sim900 como lo muestra la fig. 4.6 (b), en el circuito se encenderá un led que nos indica la activación del módulo sim900 y está listo el programa para funcionar, después activaremos la alarma mediante una botonera que prendera un led indicando la activación de la alarma como lo muestra la figura 4.6 (c), y nos mostrara en el terminal de la Raspberry pi un mensaje de que el botón fue presionado y que la alarma se encendió, procederá el módulo sim900 a mandar el mensaje, al número programado.

```
pi@raspberrypi ~/arduPi/1 $ sudo ./sms-alarma
Se activo el modem SIM900
Se inicio el programa
El boton fue presionado
La alarman se encendio
Se envio un sms por el modem SIM900
```

Figura 4.6.(b).Mensajes de la ejecución del programa en el terminal de la raspberry pi



Figura 4.6.(c).Proyecto funcionando

## CONCLUSIONES

1.- En este proyecto nos ha dado la oportunidad de aprender mucho sobre una nueva tecnología y sus diferentes aplicaciones en las cuales podremos aprovechar. Este proyecto se basa comúnmente en el uso de las Raspberry pi, con este dispositivo se pudo construir un sistema de envío de mensaje con la ayuda de una placa arduino shield y un módulo GSM. Como podemos resaltar este proyecto es una integración de diferentes tecnologías y poder lograr un mismo fin, con la cual se desarrolló nuestro proyecto.

2.- Una de las acotaciones más importantes que podemos concluir es que gracias al arduino shield se pudo acoplar el módulo GSM y poder así tener la comunicación que queríamos, para tener un resultado correcto usamos el CUTECOM para verificar si la comunicación es correcta con comandos AT y lo verificamos con una respuesta Ok.

3.-Podemos concluir que unas de las herramientas importantes fue la librería (ardupi) que hizo posible la ejecución de nuestra programación con los interfaces que hicimos en los ejercicios, ya que la función de esta librería es ejecutar cualquier programa de arduino en nuestra Raspberry pi y podremos conectar cualquier módulo de arduino sin problema desde la interfaz Raspberry pi y arduinoshield.

4.-Con este sistema nos ayudaría mucho cuando un intruso trata de ingresar en un lugar restringido, un ejemplo ingresar a una bodega sin autorización, la alarma se enciende y envía un mensaje al supervisor para indicarle que su sistema de seguridad ha sido infiltrado, nosotros podemos configurar el número en la cual se recibirá el mensaje de alerta.

## **RECOMENDACIONES**

1.- Según se avanzaba en el desarrollo del proyecto se me hacía más necesario probar los distintos dispositivos que podía instalar. Es importante tener las herramientas adecuadas para trabajar dado que en algún momento del proyecto podría fallar y tener problemas en su funcionamiento en algún dispositivo o verificando a nivel físico sus conexiones.

2.- A la hora de programar los distintos elementos ayuda mucho realizar de antemano un esquema con las funciones que necesitamos y no alterarlo ya que un proyecto cada vez va haciéndose más grande y tener que cambiar una cosa que parece insignificante pero puede dar mucho trabajo adicional.



3.-En el modulo GPRS/GSM Quadband (SIM900) es recomendable hacerlo funcionar con una fuente externa por que probablemente el RaspberryPi no es capaz de dar toda la corriente al módulosim900, utilizaremos una fuente de alimentación externa(12V-2A) en el escudo. Recuerde ajustar el interruptor del módulo sim900 EXT para hacer la alimentación externa. Ajustar la velocidad en baudios 115200bps y tener mucho en cuenta que los comandos para este módulo son con letra mayúscula.

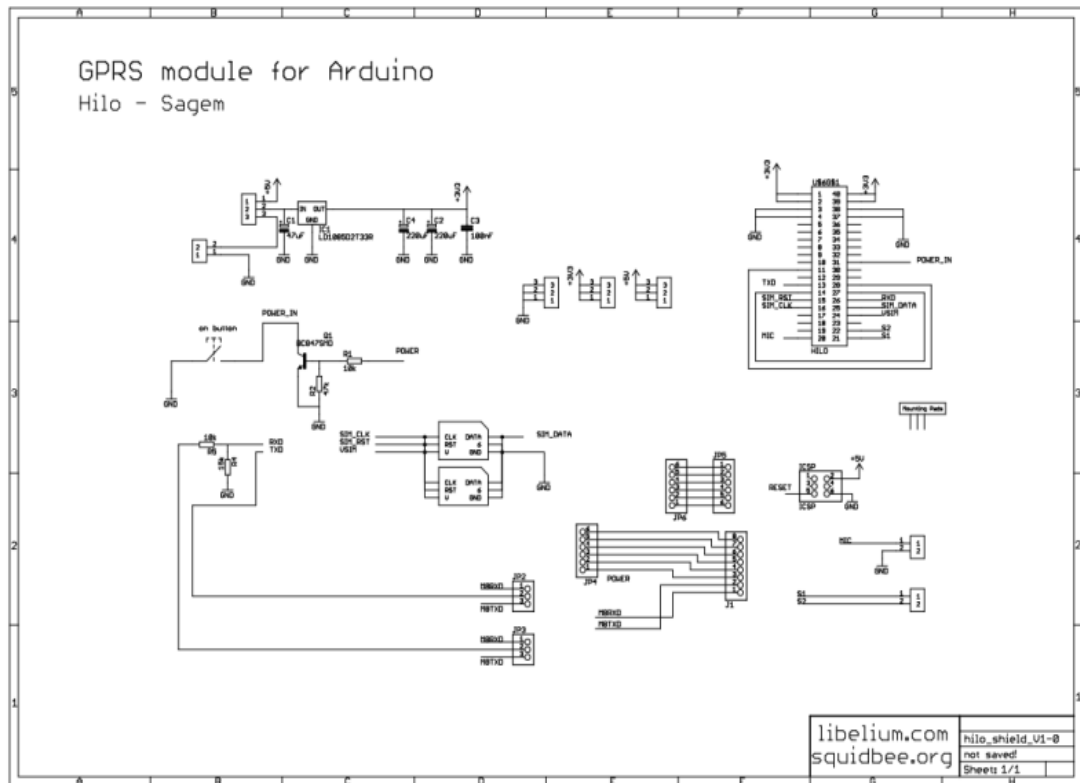
## BIBLIOGRAFÍA

- [1]Raspberry pi disponible en: <http://rayhightower.com/blog/2012/12/03/ruby-on-raspberry-pi/>
- [2,3,4] Arduino shield está disponible :<http://www.cooking-hacks.com/index.php/documentation/tutorials/raspberry-pi-to-arduino-shields-connection-bridge>
- [5]Sistema operativo Linux disponible en  
: <http://www.teinteresasaber.com/2011/03/historia-de-linux.html>
- [6] Programación en Python disponible en  
: <http://nachxs.wordpress.com/category/programacion/python/>
- [7] módulo GPRS/GSM Quadband (SIM900) está disponible  
: <http://www.cooking-hacks.com/index.php/gprs-quadband-module-sim900-for-arduino.html>
- [8] circuito de alimentación disponible: pdf sim 900 AT Command manual\_v1.05
- [9] diagrama del módulo GPRS/GSM Quadband (SIM900) está disponible  
: <http://www.cooking-hacks.com/index.php/documentation/tutorials/raspberry-pi-gprs-gsm-quadband-sim900>
- [10] Código del ejercicio 1 está disponible : en el libro Raspberry Pi Home Automation with Arduino AUTOR: ANDREW K. DENNIS Pag.116

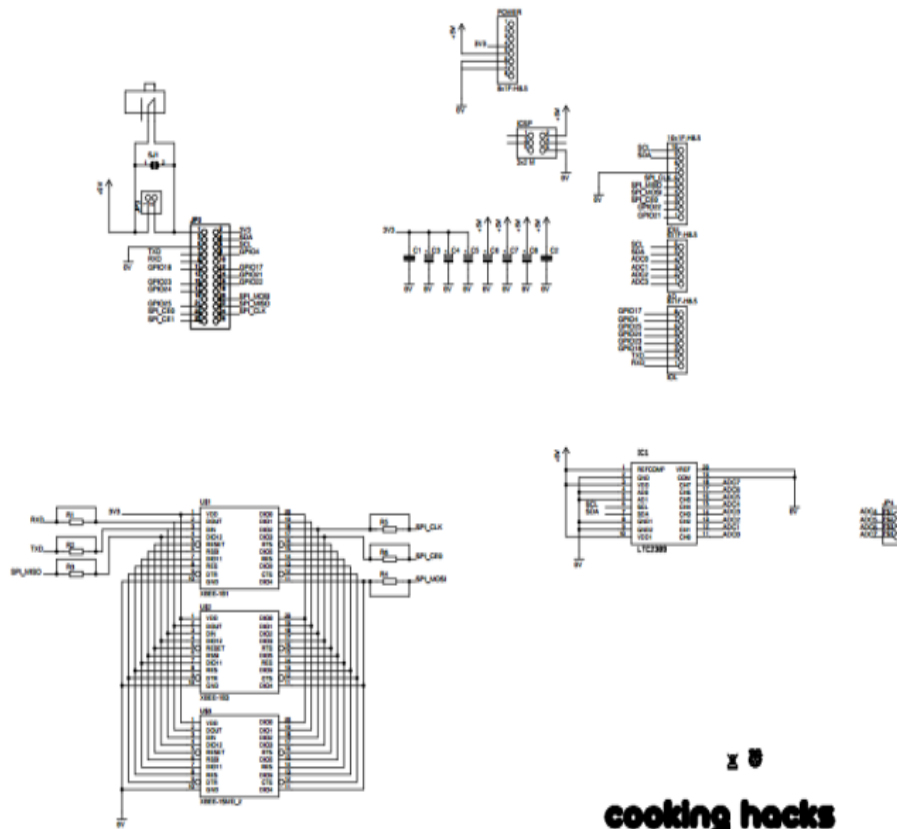
- [11] Código del ejercicio 2 está disponible: <http://www.cooking-hacks.com/index.php/documentation/tutorials/raspberry-pi-gprs-gsm-quadband-sim900>

## ANEXOS


### Esquemático módulo gprs/gsmquadband (sim900)



# Esquemático del ARDUINO SHIELD



## Gráfico de los pines de las raspberri pi

Pin 1	Pin 2	Pin 1	Pin 2	Pin 1	Pin 2
3.3 V	5 V	3.3 V	5 V		
GPI0 0	5 V	GPI0 2	5 V		
GPI0 1	GND	GPI0 3	GND		
GPI0 4	GPI014	GPI0 4	GPI014		
GND	GPI015	GND	GPI015		
GPI017	GPI018	GPI017	GPI018		
GPI021	GND	GPI027	GND		
GPI022	GPI023	GPI022	GPI023		
3.3 V	GPI024	3.3 V	GPI024		
GPI010	GND	GPI010	GND		
GPI0 9	GPI025	GPI0 9	GPI025		
GPI011	GPI0 8	GPI011	GPI0 8		
GND	GPI0 7	GND	GPI0 7		
Pin 25	Pin 26	Pin 25	Pin 26		Pin 25

**Raspberry Pi (Rev. 1)**
                         
 **Raspberry Pi (Rev. 2)**

## Menú inicial de configuración de la raspberri pi

```

Raspi-config

info          Information about this tool
expand_rootfs Expand root partition to fill SD card
overscan      Change overscan
configure_keyboard Set keyboard layout
change_pass   Change password for 'pi' user
change_locale Set locale
change_timezone Set timezone
memory_split  Change memory split
overclock     Configure overclocking
ssh           Enable or disable ssh server
boot_behaviour Start desktop on boot?
update        Try to upgrade raspi-config

                <Select>                <Finish>
  
```

## ACTIVACION DE PUERTO UART

Abra una terminal en la frambuesa, o conectarse a Raspberry Pi a través de SSH.

1. Haga una copia de seguridad del archivo / boot / cmdline.txt. `sudo cp / boot / cmdline.txt / boot / cmdline_backup.txt`

2. Editar archivo / boot / cmdline.txt: `sudo vi / boot / cmdline.txt` Este archivo contiene: `dwc_otg.lpm_enable = 0 console = ttyAMA0, 115.200 kgdboc = ttyAMA0,115200 console = tty1 $` Retire los parámetros que hacen referencia al puerto serie UART ( ttyAMA0): `dwc_otg.lpm_enable = 0 console = tty1 $`

3. Editamos la siguiente línea en / etc / inittab: `T0: 23: respawn :/ sbin / getty-L ttyAMA0 115200 vt100`

4. Reinicie la raspberry pi `sudo reboot`.

## COMPILAR BIBLIOTECA ARDUPI

Este paso se debe hacer en la RASPBERRY PI, como arduPi es una biblioteca de C++ usaremos g++ para compilarlo. Puede compilar la biblioteca arduPi obtener un archivo objeto (.o) Y utilizar este archivo para vincular su programa: `g++ -c-o arduPi.cpp arduPi.o`

Si ya ha compilado la biblioteca arduPi (paso anterior) que puede

hacer: `g++ -lrt-lpthread my_program.cpp arduPi . o-o mi_programa` Si arduPi biblioteca no se compila sólo puede compilar tanto arduPi y su programa y vincularlos en un solo paso: `g++ -lrt-lpthread my_program.cpp arduPi.cpp-o mi_programa` El `-lrt` bandera es NECESARIO porque la biblioteca utiliza el `clock_gettime` función (`time.h`). El `-lpthread` opción es necesaria porque `attachInterrupt ()` y `detachInterrupt ()` funciones utilizan hilos.

## **Ejecución del programa**

Para ejecutar el programa debe tener los permisos adecuados para utilizar GPIO (/ dev / mem necesita ser visitada en la raspberry pi). Puede ejecutar el programa con sudo: `sudo. / mi_programa`