**CONTROL DE LUCES Y VENTILACIÓN MEDIANTE ARDUINO Y SERIAL**

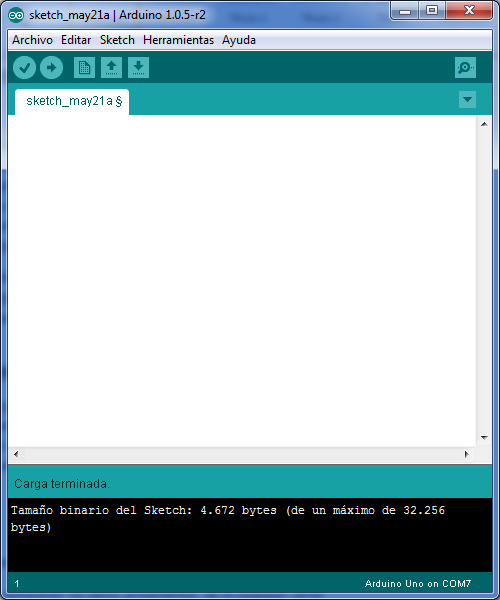
En el caso anterior controlamos las luces de las habitaciones mediante scripts BASH y los pines suministrados por la RPI, los GPIO. Pero surge un problema a la hora de utilizar estos pines para encender leds y demás, la RPI tiene una potencia de salida muy limitada y utilizar tantos componentes directamente conectado a la placa puede dañarla. Buscando alternativas encontré una forma más óptima de hacerlo y esa es utilizando el microcontrolador arduino.

Para ello tenemos que conectar la arduino mediante usb a la Raspberry, y mediante Python configurar el arduino para que reciba las ordenes de la raspberry. En raspberry tendremos que crear un script Python para mandarle las ordenes a la arduino.

**CONFIGURANDO ARDUINO**

El microcontrolador que voy a utilizar se llama arduino, podréis ver sus especificaciones en el apartado de descripción de materiales. Para cargar el esquema que contendrá el código en C, vamos a utilizar el programa suministrado por arduino, se puede bajar desde aquí <http://arduino.cc/en/Main/Software>.

Una vez instalado y abierto nos saldrá una pantalla así:



Tendremos que escribir el código C que recepcionará las ordenes provenientes de la Raspberry y en función de eso mandará señal o no a los pines correspondientes.

En este proyecto voy a utilizar la arduino para manejar 4 pines:

Pin 12 🡪 control de la luz del salón.

Pin 11 🡪 Control de la luz del exterior.

Pin 10 🡪 Control de la luz del garaje.

Pin 9 🡪 Control de la ventilación.

Sabiendo los pines que tenemos que controlar vamos a cargar este código C en la arduino:

*// Buffer para almacenar los datos provenientes de la conexion serial*

*String inData;*

*int cont = 0;*

*//diferentes habitaciones que se pueden encender correspondiendo a los pines que le vamos a asignar*

*int salon = 12;*

*int exterior = 11;*

*int garaje = 10;*

*int ventilador = 9;*

*void setup() {*

*pinMode(salon, OUTPUT);*

*pinMode(exterior, OUTPUT);*

*pinMode(garaje, OUTPUT);*

*pinMode(ventilador, OUTPUT);*

*Serial.begin(9600);*

*Serial.println("Conexion serial iniciada, esperando orden");*

*}*

*//bucle principal*

*void loop() {*

*while (Serial.available() > 0)*

*{*

*//leyendo los caracteres enviados por serial*

*char recieved = Serial.read();*

*inData += recieved;*

*cont += 1;*

*// cuando haya leido 3 caracteres se procesara la cadena*

*if (cont == 3)*

*{*

*// aqui se evaluaran las cadenas y dependiendo de la orden se modifican los estados de los pines correspondientes*

*if (inData == "H12") {*

*digitalWrite(salon, HIGH);*

*} else if (inData == "L12"){*

*digitalWrite(salon, LOW);*

*} else if (inData == "H11"){*

*digitalWrite(exterior, HIGH);*

*} else if (inData == "L11"){*

*digitalWrite(exterior, LOW);*

*} else if (inData == "H10"){*

*digitalWrite(garaje, HIGH);*

*} else if (inData == "L10"){*

*digitalWrite(garaje, LOW);*

*} else if (inData == "H09"){*

*digitalWrite(ventilador, HIGH);*

*} else if (inData == "L09"){*

*digitalWrite(ventilador, LOW);*

*}*

*inData = ""; // limpiamos el buffer y inicializamos el contador*

*cont = 0;*

*}*

*}*

*}*

***FIN DEL SCRIPT***

Para cargarla solo tendremos que escribir el código y pulsa la tecla “cargar”, si todo resulta bien nos saldrá un aviso diciéndonos que el script se ha cargado correctamente.

**CONFIGURACIÓN DE LA RASPBERRY.**

En este momento tendremos el microcontrolador arduino preparado y a la espera de las respectivas órdenes para ejecutar el encendido de las luces correspondientes, pero para mandar esas órdenes que el arduino tanto espera, hay que crear un script en python en la Raspberry que mande por serial la orden que queramos que ejecute. Para esta tarea tendremos que tener la raspberry conectada a internet ya que tendremos que instalar el paquete intérprete de python.

**sudo apt-get install python-serial**

Una vez instalado procedemos a crear el script ejecutador en la carpeta gpioscripts ubicada en el escritorio.

**Nano gpioscripts/arduino.py**

Y aquí tenemos el código del script:

#se importan la libreria pyserial y la sys

import serial

import sys

#-- Se abre el puerto correspondiente a la arduino por usb y se establece

# el tiempo de espera maximo

try:

s = serial.Serial('/dev/ttyACM0', 9600)

s.timeout=2

except serial.SerialException:

print "Error al abrir el puerto."

sys.exit()

# Se envia la orden a la arduino mediante serial

orden = sys.argv[1]

s.write(orden)

**FIN DEL SCRIPT**

¡Ya estamos a un paso de conseguir lo que nos proponemos!

Antes de mostrar cómo se ejecuta este script hay que tener en cuenta los parámetros que tenemos que mandarle para encender las diferentes luces:

**H12🡪** Significa “HIGH” pin 12, lo que viene siendo encender pin12

**L12 🡪**Significa “LOW” pin 12, lo que viene siendo apagar pin 12

Sabiendo esto y recordando las correspondencias de los pines con las habitaciones:

Pin 12 🡪 control de la luz del salón.

Pin 11 🡪 Control de la luz del exterior.

Pin 10 🡪 Control de la luz del garaje.

Pin 9 🡪 Control de la ventilación.

Podemos ejecutar el script desde el escritorio de la siguiente manera:

**Python arduino.py H12 (si queremos encender la luz del salón)**

**Python arduino.py L12 (Si por el contrario queremos apagarla)**

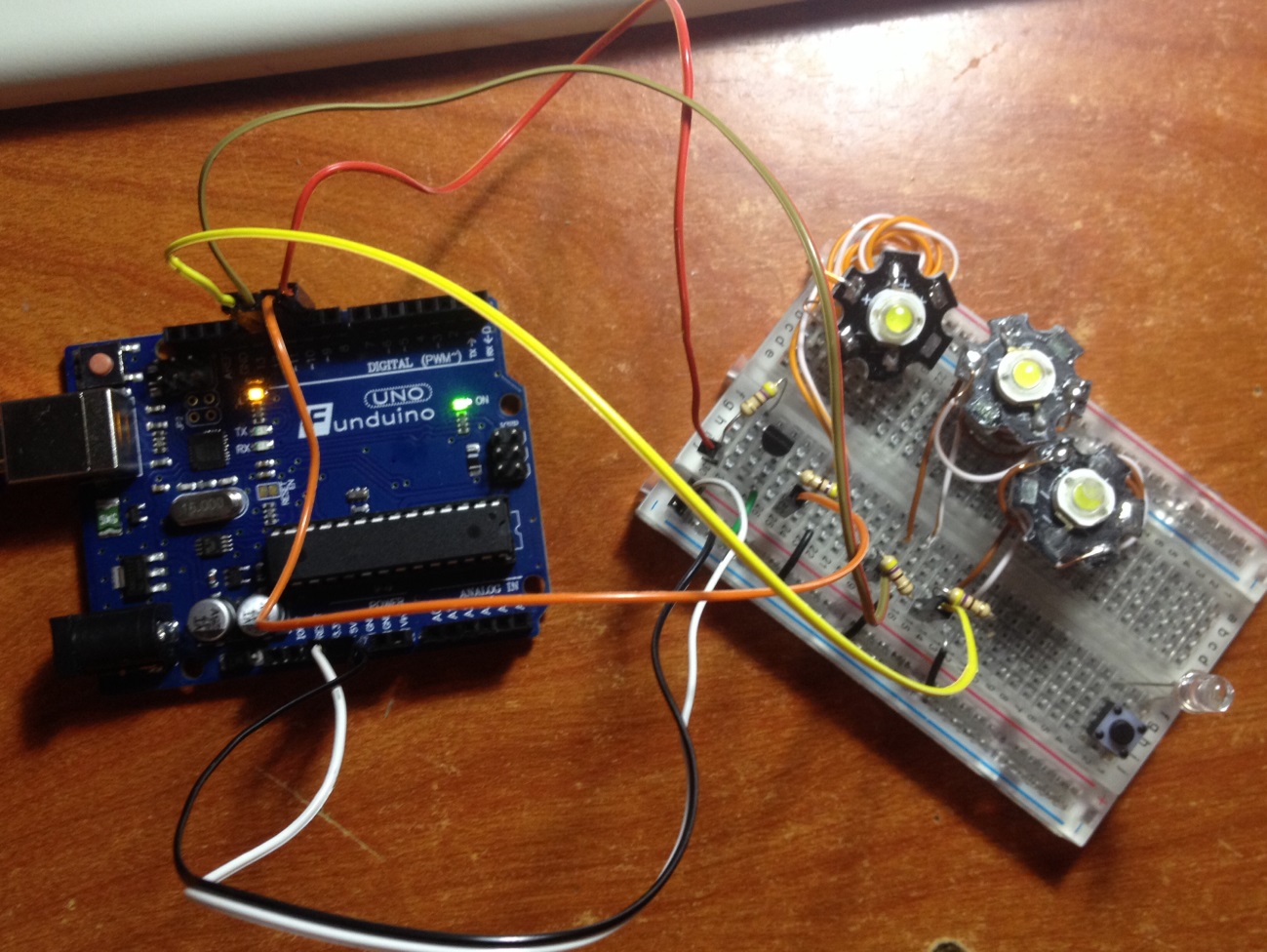
**PROBLEMAS**

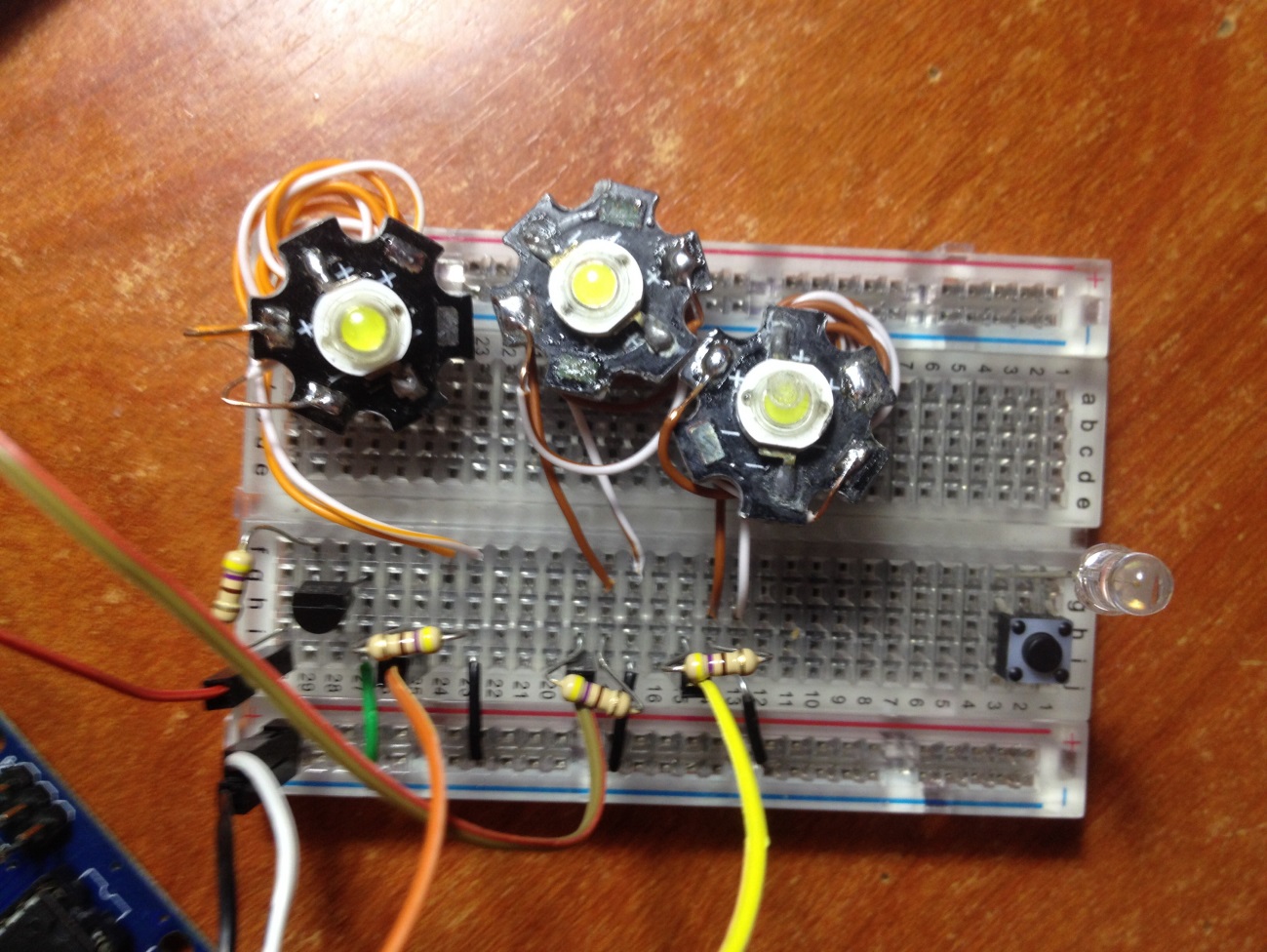
El problema de esta configuración, es que cuando se conecta por serial a la arduino UNO, que es la que estoy utilizando, esta se reinicia debido a que está configurada así de fábrica para poder cargarle los sketch sin tener que pulsar el botón de reset. La única solución para esta placa es quitando una soldadura para deshabilitar el reinicio al conectar por serial. Como no quiero estropear la placa no voy a optar por esta opción como configuración final. Cuando mandas una orden por serial la arduino la recibe y enciende los leds que precises, pero cuando mandas otra vez una orden para encender otros o simplemente para apagarlos, esta se reinicia y hace parpadear los leds, cosa que no quiero que mi proyecto haga. Para ello voy a utilizar otra configuración que veremos en el siguiente documento.

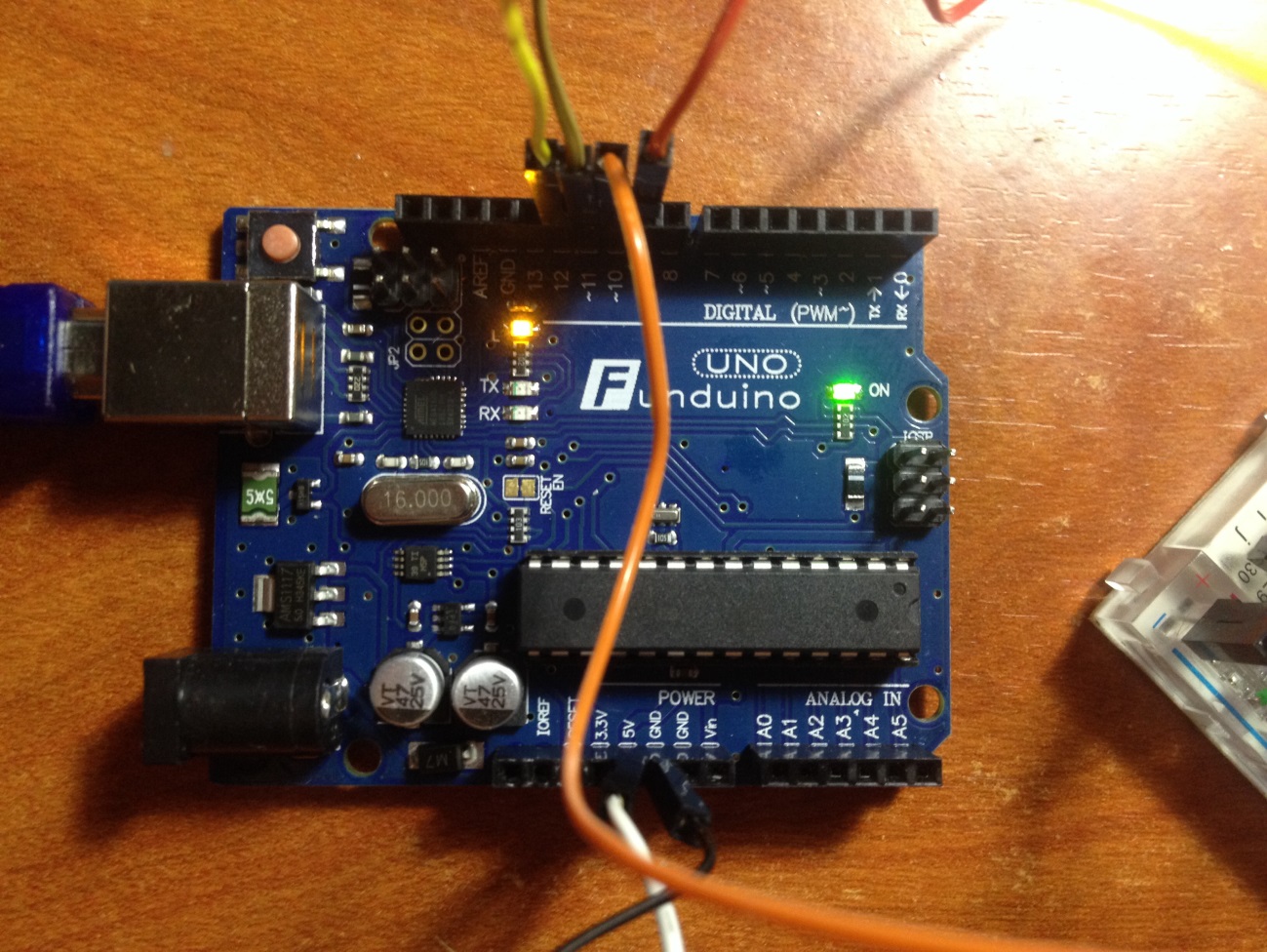
**Solución al reinicio automático**

Finalmente para solucionar el problema del reinicio automático cuando se establece comunicación serial. Se coloca un condensador de 10 microfaradios entre la patilla de reset y la de tierra. Con esto se hace más viable esta forma de controlar la arduino, ya que precisa de menos cableado.

**CONEXIÓN DE CABLEADO EN LA ARDUINO.**

****

****

****

Fuentes: <http://geekytheory.com/arduino-raspberry-pi-raspduino/>, <http://pyserial.sourceforge.net/index.html>, <http://stackoverflow.com/questions/5697047/convert-serial-read-into-a-useable-string-using-arduino>, <http://www.comoprogramar.org/ejemplos-de-python-codigo/>.