

Talleres en UPAYAKUWASI - Cayambe, Ecuador

Contenido

1. Introducción de los espacios (upayakuwasi-un/loquer)
2. Presentación de participantes y expectativas
3. Uso y adaptación a recursos naturales (Caminata por UPAYAKUWASI)
4. Introducción a electrónica, sensores y actuadores
5. Control de Entradas y Salidas
6. ESP8266
7. Arduino
8. Sensor humedad
9. Sensor temperatura y humedad
10. Sensor luz
11. Actuador relevo
12. Práctica

Taller Mujeres

<http://piratepad.net/unloquerupayakuwasi>

1. Taller en 4 capas
2. Riego, Automatización Válvulas, Sensores, Plataforma
3. Primera capa: (Analógico) Ellas dibujan sus huertos, ¿de donde sacan el agua?, ¿cuáles válvulas prenden?, ¿A qué horas?, presentan al resto
4. Segunda capa: (Automatización abrir cerrar válvulas) Se programan tiempos para abrir válvulas
5. Tercer capa: (¿qué pasa si llueve?, necesito saber que ya no tengo que regar) Cuáles son los sensores, ¿cómo se programan?
6. Cuarta capa: (Afinación y plataforma de control)

¿Qué es lo que se llevan? → Relevo con ESP8266

Cada etapa tiene que tener su presupuesto.

Conexión de wifi para raspberrypi vía esp8266

Por medio del puerto serie <http://pwiatrowski.com/technology/raspberry-pi-zero-esp8266-internet/>
<https://github.com/jeelabs/esp-link>

Por medio de SPI <https://oshlab.com/esp8266-raspberry-pi-gpio-wifi/> Proyecto original
<https://hackaday.io/project/8678/instructions>



Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I2C)		DC Power 5v	04
05	GPIO03 (SCL1 , I2C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)		(I2C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40





Para instalar el driver en archlinux-arm nos basamos en el paquete <https://aur.archlinux.org/packages/esp8089-git/> pero se modifica el PKGBUILD como se muestra a continuación:

```
# Maintainer: Swift Geek
# TODO: DKMS

_gitname=esp8089
pkgname=$_gitname-git
pkgver=2016.08.07
pkgrel=1
pkgdesc="Linux kernel module driver for the ESP8089 WiFi chip"
arch=('i686' 'x86_64' 'armv7h' 'armv6h')
url="https://github.com/al177/$_gitname"
license=('GPL')
install=$_gitname.install
depends=('linux')
makedepends=('git' 'linux-headers')
options=(!strip)
source=("git+${url}.git")

md5sums=('SKIP')

pkgver() {
    cd "$srcdir/$_gitname"
    git log -1 --format="%cd" --date=short | sed 's|-|.|g'
}

prepare() {
    sed -i s/RX_FLAG_HT/RX_ENC_HT/ esp8089/esp_sip.c
    sed -i s/RX_FLAG_SHORT_GI/RX_ENC_FLAG_SHORT_GI/ esp8089/esp_sip.c
}

build() {
    cd "$srcdir/$_gitname/"
    make modules M=../$_gitname CONFIG_ESP8089=m
    gzip -f esp8089.ko
}
```

```
package() {
    cd "$srcdir/$_gitname/"
    install -Dm644 esp8089.ko.gz "$pkgdir/usr/lib/modules/$(uname -
r)/kernel/drivers/net/wireless/esp8089.ko.gz"
    #depmod -a $(uname -r)
}
```

una vez instalado el paquete se instala el módulo con `sudo modprobe esp8089`

Para configurar la red inalámbrica se usa `netctl` como dice acá

<https://raspberrypi.stackexchange.com/questions/7987/wifi-configuration-on-arch-linux-arm#7992>

```
/etc/netctl# install -m640 examples/wireless-wpa wireless-home
/etc/netctl# cat wireless-home
Description='A simple WPA encrypted wireless connection'
Interface=wlan0
Connection=wireless
Security=wpa

IP=dhcp

ESSID='MyNetwork'
# Prepend hexadecimal keys with \"
# If your key starts with ", write it as '\"<key>\"'
# See also: the section on special quoting rules in netctl.profile(5)
Key='WirelessKey'
# Uncomment this if your ssid is hidden
#Hidden=yes
```

Luego arranque el servicio # `netctl start wireless-home`



Tenemos problemas con la asignación de dirección ip por dhcpd, relacionado con timeouts

<https://wiki.archlinux.org/index.php/Netctl#Troubleshooting>

<https://bbs.archlinux.org/viewtopic.php?pid=1399842#p1399842>

<https://archlinuxarm.org/forum/viewtopic.php?f=31&t=5424>



RTC digital por medio de la libreria Time para ESP8266

Básicamente la libreria funciona por alarmas y temporizadores. Una **Alarma** es una tarea que ocurre a determinada hora del día. Mientras que un **temporizador** es una tarea programada que ocurrirá **después** de que haya pasado algún intervalo de tiempo.

[repositorio](#)





Muy importante recordar que en vez de usar el `delay(algunvalor)` de Arduino, se debe usar el de la librería `alarm`. De esta manera: `Alarm.delay(algunvalor)`

```
#include <Time.h>
#include <TimeAlarms.h>

int pin = D1;
int hora, minutos;
int alarma;

void setup() {
  Serial.begin(9600);
  hora = 8;
  minutos = 1;
  //fijamos el tiempo el Domingo 25 de marzo del 2018 a las 1:40 pm
  setTime(hora,0,0,3,25,18);
  //Creamos las alarmas
  Alarm.alarmRepeat(hora,minutos,0,prenderSensorRiegoManana); // 8:30am
  cada día
  Alarm.alarmRepeat(hora,minutos + 3,0,apagarSensorRiegoManana); // 8:30am
  cada día

  //Alarm.alarmRepeat(17,45,0,EveningAlarm); // 5:45pm cada día
  //Alarm.alarmRepeat(dowSaturday,8,30,30,WeeklyAlarm); // 8:30:30 cada
  sabado

  //alarma = 15;
  //Alarm.timerRepeat(alarma, Repeats); // Timer cada 15 segundos
  //Alarm.timerOnce(10, OnceOnly); // Llamado una vez despues de 10
  segundos
  pinMode(pin, OUTPUT);
  digitalWrite(pin, LOW);
}

void loop() {
  digitalClockDisplay();
  Alarm.delay(1000);
}

// funciones para llamar cuando se dispara una alarma
void prenderSensorRiegoManana(){
  Serial.println("Alarm: - turn lights on");
  digitalWrite(pin, HIGH);
}

void apagarSensorRiegoManana(){
  Serial.println("Alarm: - turn lights off");
  digitalWrite(pin, LOW);
}
```

```
}

void EveningAlarm(){
  Serial.println("Alarm: - turn lights on");
}

void WeeklyAlarm(){
  Serial.println("Alarm: - its Monday Morning");
}

void ExplicitAlarm(){
  Serial.println("Alarm: - this triggers only at the given date and time");
}

void Repeats(){
  Serial.println("Activo la Alarma");
  digitalWrite(pin, HIGH);
  Alarm.delay(5000);
  digitalWrite(pin, LOW);
  Alarm.delay(1000);
}

void OnceOnly(){
  Serial.println("This timer only triggers once");
}

void digitalClockDisplay()
{
  // digital clock display of the time
  Serial.print(hour());
  printDigits(minute());
  printDigits(second());
  Serial.println();
}

void printDigits(int digits)
{
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}

void prendaSensor(){
  pinMode(pin, HIGH);
}

void apagueSensor(){
  pinMode(pin, LOW);
}
```

Código para activar algo desde una página web con ESP8266

Este código genera una página web para controlar el encendido o apagado de algún dispositivo.

```
#include <ESP8266WiFi.h>

const char* ssid = "UPAYAKUWASI";
const char* password = "nosotrxs";

int ledPin = D1; // GPIO13
WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  delay(10);

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);

  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address
  Serial.print("Use this URL to connect: ");
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/");
}

void loop() {
```

```
// Check if a client has connected
WiFiClient client = server.available();
if (!client) {
  return;
}

// Wait until the client sends some data
Serial.println("new client");
while(!client.available()){
  delay(1);
}

// Read the first line of the request
String request = client.readStringUntil('\r');
Serial.println(request);
client.flush();

// Match the request

int value = LOW;
if (request.indexOf("/LED=ON") != -1) {
  digitalWrite(ledPin, HIGH);
  value = HIGH;
}
if (request.indexOf("/LED=OFF") != -1) {
  digitalWrite(ledPin, LOW);
  value = LOW;
}

// Set ledPin according to the request
//digitalWrite(ledPin, value);

// Return the response
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println(""); // do not forget this one
client.println("<!DOCTYPE HTML>");https://processing.org/download/
client.println("<html>");

client.print("<h1>Su dispositivo ahora esta: </h1>");

if(value == HIGH) {
  client.print("On");
} else {
  client.print("Off");
}
client.println("<br><br>");
client.println("<a href='\"/LED=ON\"'><button>Prender </button></a>");
client.println("<a href='\"/LED=OFF\"'><button>Apagar </button></a><br
/>");
client.println("</html>");
```

```
delay(1);  
Serial.println("Client disonnected");  
Serial.println("");  
  
}
```

From:

<https://wiki.unloquer.org/> -

Permanent link:

https://wiki.unloquer.org/proyectos/talleres_esp/upayakuwasi

Last update: **2018/03/26 18:19**

