Control de acceso mediante NFC con Arduino

Miguel Viñas Gutiérrez
Grado de Tecnologías de Telecomunicación
TFG - Arduino

Oriol Jaumandreu Sellarès
Pere Tuset Peiró

01/2017

FICHA DEL TRABAJO FINAL

Título del Trabajo:	**Control de acceso mediante NFC con Arduino**
Nombre del autor:	**Miguel Viñas Gutiérrez**
Nombre del consultor:	**Oriol Jaumandreu Sellarès**
Nombre del PRA:	**Pere Tuset Peiró**
Fecha de entrega:	**01/2017**
Titulación:	**Grado de Tecnologías de Telecomunicación**
Área del Trabajo final:	**TFG - Arduino**
Idioma del trabajo:	**Castellano**
Palabras clave:	**Arduino, NFC, acceso**

Resumen del Trabajo:

En el siguiente trabajo se desarrolla un sistema de control de acceso para implementar en una cadena de tiendas. El núcleo central o dispositivo principal del trabajo será realizado con un controlador Arduino.

El sistema desarrollado incluye un acceso 'sin contacto' mediante llavero NFC, avisador visual y sonoro de acceso correcto o fallido, una pantalla LCD para mostrar información al usuario, conexión Ethernet y wifi, gestión de los accesos en una base de datos y la posibilidad de visualizar esos datos en un navegador de Internet mediante una aplicación web HTML/PHP/CSS.

Abstract:

The following work develops an access control system to implement in a chain of stores. The core or main device for making it happen will be an Arduino controller.

The developed system includes a contactless access via an NFC key chain, visual and audible warning of correct or failed access, an LCD to display information to the user, Ethernet and Wi-Fi connection, access management in a database and the possibility to display that data in an Internet browser using an HTML/PHP/CSS web application.

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. Contexto del Trabajo

Cada vez es más habitual que las empresas coloquen dispositivos de control de acceso o sistemas de fichaje para sus trabajadores. Dentro de estos sistemas podemos encontrar opciones de mucha calidad y precios desorbitados, completamente lejos del alcance de las pequeñas empresas.

Con este sistema de control de acceso basado completamente en Arduino se pretende demostrar que un sistema fiable, robusto y no necesariamente caro (menos de 100€ por acceso) puede ser un sistema más que válido para una empresa que desee controlar la hora de entrada y salida de sus trabajadores.

El jefe o empresario de esta pequeña empresa podrá recibir avisos que le indiquen cuándo comienzan la jornada sus trabajadores, cuándo la terminan e incluso poder llevar un registro mensual de sus accesos, así como la suma de las horas trabajadas por cada trabajador.

1.2. Justificación

En la empresa en la que se pretende instalar el sistema (GoblinTrader SL), el salario por hora de un trabajador de tienda es de 7€, por tanto una entrada al trabajo 10 minutos tarde durante 6 días de un solo trabajador supone unas pérdidas a las empresa de 7€. Si se cuenta con que en la empresa hay 32 trabajadores las pérdidas de dinero debido a los retrasos o a las salidas prematuras pueden ser abultadas.

Además , el poder llevar la cuenta de las horas totales supone un ahorro de personal que controle las horas trabajadas por cada trabajador, ya que el sistema guardará los datos para su visualización y análisis más cómodo y rápida.

En el mercado existen actualmente multitud de fabricantes que comercializan sistemas de control de acceso. En este primer ejemplo, se observa un control de acceso de gama baja con un precio también muy bajo:

Marca: Fabricante de marca blanca chino. Muchas marcas vendedoras

Precio: 23 € IVA incluido

Funcionalidades: sensor NFC y teclado numérico. Indicador LED

Deficiencias: no dispone de almacenamiento de datos ni envío de los mismos a un sistema externo. No dispone de avisador gráfico con el nombre de la persona que se está registrando ni avisa de la causa de errores.

Figura 1

El siguiente ejemplo, en cambio, es un sistema integral de gama media/alta con un precio mucho más elevado y funcionalidad completa:

Marca: Safescan

Precio: 434 € IVA incluido

Funcionalidades: sensor NFC, teclado numérico, pantalla LCD y almacenamiento en base de datos de los accesos registrados. Software de gestión.

Deficiencias: no dispone de envío mediante sistema inalámbrico, por lqo ue debe estar conectado en todo momento a una red LAN.

Figura 2

1.3. Objetivos del Trabajo

La siguiente lista enumera los objetivos que se pretenden alcanzar al finalizar el proyecto. El orden de la lista indica el orden de importancia de los objetivos:

1. Se pretende realizar un montaje para controlar un acceso a un lugar determinado mediante un microcontrolador Arduino y una placa NFC[1].

2. Se buscará integrar en el sistema diversos avisos para confirmar el acceso o para denegarlo. Estos avisos serán un aviso luminoso mediante LED[2] , un aviso sonoro y un sistema visual mediante una pantalla LCD[3].

3. Como medida de seguridad, aparte de la tarjeta/llavero NFC, que será personal, se pretende instalar un teclado numérico para que la persona pueda introducir un número secreto, teniendo así dos maneras diferentes de acceso.

4. Se instalará un módulo Ethernet para comunicar el sistema con Internet. Esto permitirá guardar los datos de acceso registrados en el sistema. Como mejora a este sistema, se instalará una conexión wifi (conexión inalámbrica).

5. Como última medida de seguridad se pretende instalar una cámara de fotos que saque una foto cada vez que se permite un acceso al lugar.

6. Todos estos datos serán enviados a un servidor externo mediante la conexión wifi del objetivo número 4 y almecenados en una base de datos.

7. El objetivo final será organizar todos los componentes del sistema en una caja cerrada, evitando el cableado visto.

Hablando de la propia empresa donde se instalará el sistema, el objetivo será llevar un control exacto de las horas trabajadas por sus empleados, para así disminuir costes y analizar las horas extras realizadas por todos los trabajadores en la empresa.

1.4. Enfoque y método seguido

Para el desarrollo del Trabajo, se ha seguido un método por etapas que simplifica la implementación de nuevos dispositivos en el sistema. Cada dos nuevos componente introducidos en el sistema disponen de un epígrafe (etapa) propio.

Cada una de las etapas se ha dividido en tres partes bien diferenciadas: *Montaje*, *Código* y *Pruebas*.

En el Montaje se especifican los componentes utilizados, su conexión con el resto del sistema y se muestra un esquema de conexión realizado con el software gratuito

[1] *Near Field Communication*
[2] *Light-emitter Diode*
[3] *Liquid Crystal Display*

Fritzing[4].

En la parte de Código se muestra y explica el código de Arduino utilizado para hacer funcionar la parte del sistema. Para ello se utiliza el IDE oficial de Arduino[5]. El código completo del Trabajo se adjunta en la carpeta del proyecto.

En la parte de Pruebas se comprueba que el sistema responde a las acciones tal y como se planteó en los objetivos y en la Planificación del Trabajo (Ver punto 1.4).

1.5. Planificación del Trabajo

En el desarrollo descrito a continuación se han dividido los objetivos propuestos (Ver 1.2.) en pequeñas tareas que serán la base del Trabajo, cada una de ellas a modo de hito que se debe ser superado antes de pasar al siguiente.

Las tareas también se han dividido en dos grandes grupos, las Tareas Principales (de la 1 a la 8) y las Tareas Secundarias (de la 9 a las 11). Estas últimas son tareas opcionales que dependen de la consecución de las Tareas Principales para ser realizadas.

Cada tarea incluye al lado de su nombre una estimación de tiempo para realizarla y un número de PEC (Prueba de Evaluación Continua) en la que deberá estar finalizada.

Todas las tareas se describen a continuación:

1. Realizar un montaje básico del sistema Arduino (PEC 2 - 2 horas)

a. Adquirir los componentes descritos en los objetivos. Inicialmente serán un dispositivo Arduino, una protoboard, cables, 2 LEDs, 2 resistencias de 220Ω, una tarjeta sensor NFC, una tarjeta NFC y un llavero NFC.

b. Realizar el esquema de conexión del sistema básico.

c. Montar el sistema guiado por el esquema.

d. Instalar el IDE de Arduino y sus drivers si es necesario.

2. Acceso mediante tarjeta NFC o mediante llavero NFC (PEC 2 - 2 horas)

a. Escribir el código que permita encender un LED verde si el sistema ha leído la tarjeta correcta o que encienda un LED rojo si no lo ha hecho.

b. Realizar las pruebas con la tarjeta y el llavero.

3. Añadir al sistema un indicador sonoro con buzzer pasivo para indicar si la lectura de la tarjeta/llavero se ha realizado correctamente (PEC 2 - 1 hora)

[4] *http://fritzing.org/home/*
[5] *https://www.arduino.cc/en/Main/Software*

a. Una vez que el sistema permita o no el acceso, se instalará un pequeño altavoz en la protoboard, conectado a una salida del Arduino.

b. Si la tarjeta correcta es leída, el sistema emitirá un sonido de aviso.

4. Añadir al sistema una pantalla LCD para indicar a la persona que desea acceder si el acceso o la salida se ha realizado correctamente (PEC 2 - 6 horas)

a. Conectar una pantalla LCD de 2 líneas de 16 caracteres.

b. Investigar sobre las diversas librerías que existen para este dispositivo.

c. La pantalla informará, con el nombre del trabajador, si este se ha registrado correctamente o no.

5. Añadir al sistema un teclado numérico para permitir a la persona que quiere acceder introducir un pin en vez de usar el contacto NFC (PEC 3 - 6 horas)

a. Conectar al sistema un teclado numérico.

b. Asignar un PIN de 4 dígitos a cada trabajador.

c. Escribir el código que permita introducir el PIN. Si es correcto, se quedará registrado el acceso. Si no, el sistema informará del error.

6. Añadir al sistema una tarjeta Ethernet para poder comunicar el Arduino con Internet y configurar el sistema para que guarde los datos de acceso en una base de datos (PEC 3 - 12 horas)

a. Conectar un Shield Ethernet al sistema.

b. Escribir el código que permita la conexión del Arduino con Internet.

c. El sistema debe mostrar en el Monitor Serial la dirección IP del dispositivo Arduino una vez que un usuario se ha registrado correctamente. Con esto se comprobará la conexión en red con el Router.

b. Crear una base de datos mySQL con los campos necesarios.

c. Escribir el código que enlace el Arduino con la base de datos.

e. Probar accediendo varias veces con la tarjeta para ver si los datos se escriben correctamente en la base de datos.

7. Añadir al sistema una tarjeta WiFi para sustituir al Ethernet como sistema de comunicación con Internet (PEC 3 - 6 horas)

a. Retirar el Shield Ethernet y conectar un adaptador WiFi.

b. Programarlo para que funcione de la misma manera que el Ethernet pero siendo completamente inalámbrico.

8. Construir una caja para poder instalar el sistema (PEC 3 - 3 horas)

a. Una vez montados todos los componentes hasta este punto, se fabricará una caja para poder meterlo todo, que disponga de salidas para los puertos y que deje ver la pantalla LCD en el exterior.

9. Crear una sencilla aplicación web para acceder al historial de los datos de acceso (extra - 6 horas)

Tarea Secundaria que depende del desarrollo de las Tareas Principales.

10. Añadir al sistema una cámara para que al autorizar un acceso, saque una foto y la almacene en una tarjeta microSD (extra - 6 horas)

Tarea Secundaria que depende del desarrollo de las Tareas Principales.

11. Permitir que las fotografías sacadas en cada acceso puedan guardarse en un servidor externo y visualizadas mediante la aplicación web del punto 10 (extra - 8 horas)

Tarea Secundaria que depende del desarrollo de las Tareas Principales.

El diagrama de Gantt que muestra los tiempos planeados para el desarrollo de cada objetivo es el siguiente:

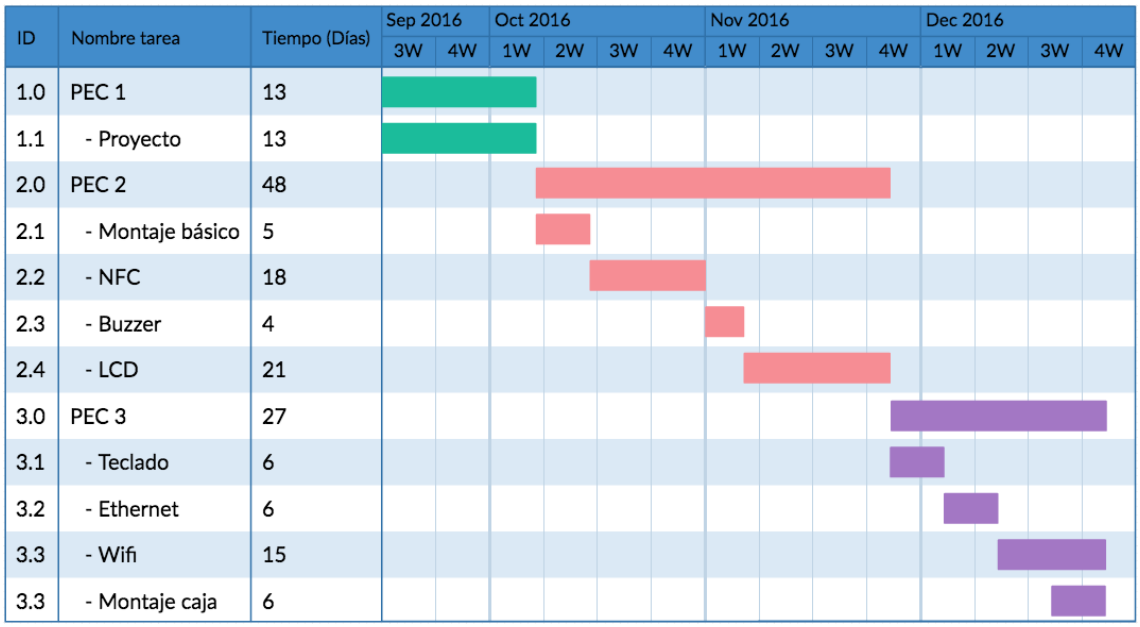


Figura 3

1.6. Breve sumario de productos obtenidos

Al finalizar el proyecto se ha conseguido construir y programar un sistema autónomo para gestionar el acceso de entrada y salida a un comercio u oficina. El sistema detecta tarjetas y llaveros NFC, avisa al usuario de la entrada correcta o fallida mediante una pantalla LCD, una viso luminoso y un aviso sonoro y envía los datos mediante cable o wifi a una base de datos externa.

Los datos en dicha base de datos puede ser visualizados a través de un navegador de Interner y filtrados según trabajador o rango de fechas del acceso.

2. MONTAJE BÁSICO Y DETECCIÓN NFC

2.1. Montaje

El primer paso en el montaje del sistema básico es adquirir los componentes que servirán como base para el resto del Trabajo. Se utiliza un dispositivo basado en el Arduino Leonardo de la marca Elekfreaks[6].

El elemento de detección NFC es un RFIO-RC522 genérico, un dispositivo muy estandarizado y con multitud de soporte en las comunidades de Internet. Junto con el dispositivo vienen incluidos un llavero NFC y una tarjeta NFC.

Completan los componentes 2 LED's estándar rojo y amarillo, 2 resistencias de 1/4W 220Ω, una protoboard tamaño pequeño y diversos cables de 0,5mm^2.

A la hora del montaje hay que prestar especial atención a las conexiones del Arduino Leonardo, bastante diferentes a las del más habitual, Arduino Uno. En este caso, el módulo de detección NFC no va conectado a los pines habituales, si no a un socket especial que dispone el Arduino Leonardo. Por tanto, los pines del detector SDA, RST, SCK, MOSI y MISO van conectados a esos mismos pines del Arduino, como puede verse en el siguiente esquema de conexión:

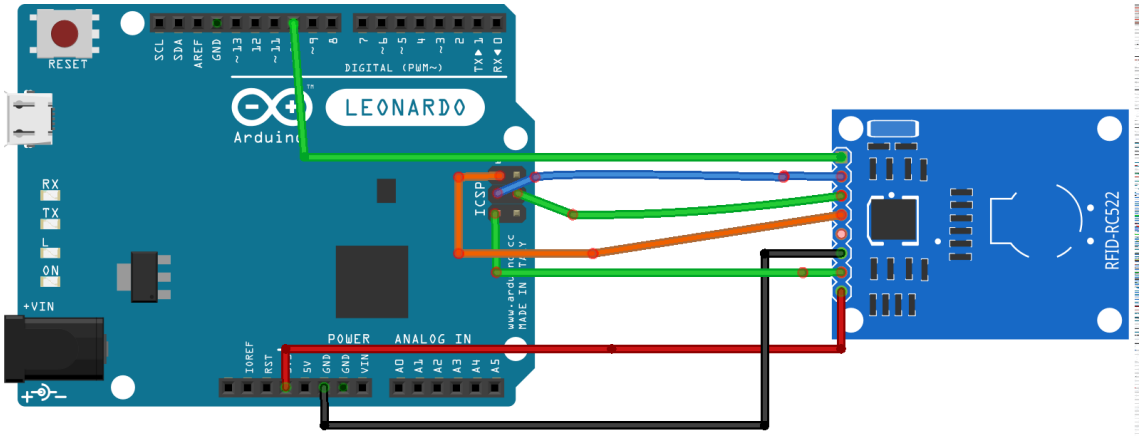


Figura 4

La conexión del sistema de aviso visual mediante LED se realiza mediante los pines de salida digitales 2 y 3.

Como se observa, los LED se conectan directamente en la protoboard, pero el módulo de detección NFC va conectado al sistema directamente con los cables, sin estar ubicado en la protoboard.

La conexión del sistema básico incluyendo el aviso visual con LED se muestra a continuación:

[6] *https://www.elecfreaks.com/estore/*

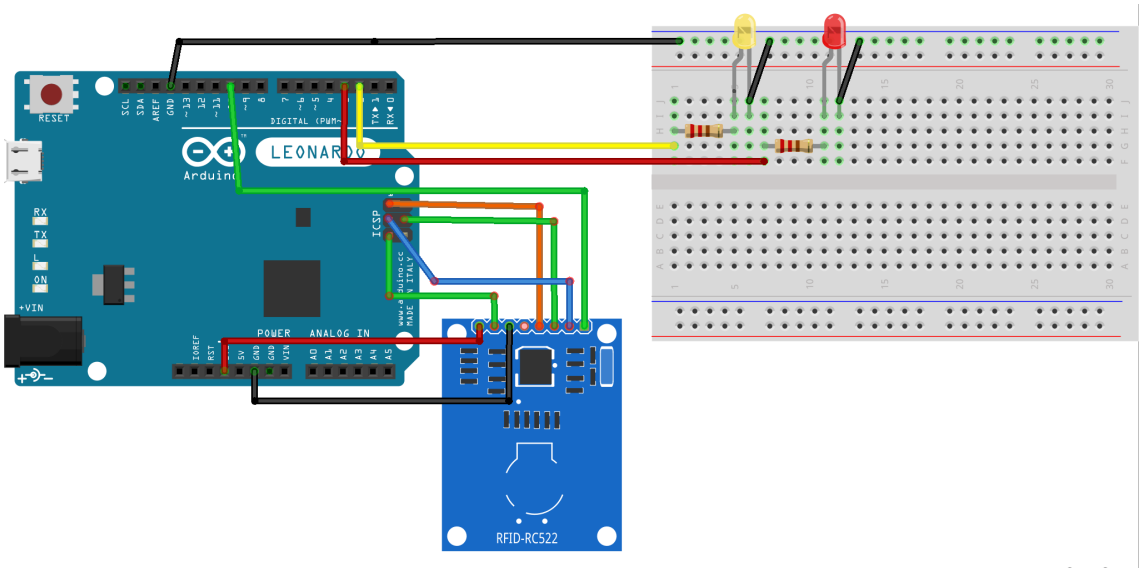


Figura 5

2.2. Código

El código completo del Trabajo se adjunta en la carpeta del proyecto.

2.3. Pruebas

En las pruebas se observa que el sistema funciona correctamente, habiéndose cumplido los objetivos de las siguientes tareas:

1. Realizar un montaje básico del sistema Arduino

2. Acceso mediante tarjeta NFC o mediante llavero NFC (y avisos LED)

Como de momento no existe otro sistema visual que no sea el sistema mediante LED's, los avisos visuales de texto se muestran a través del Serial Monitor (de ahora en adelante, solo Monitor) del propio IDE de Arduino. Al abrirlo se observa un mensaje que insta a aproximar el llavero o tarjeta al lector.

Una vez que el usuario lo ha realizado, un nuevo mensaje a través del Monitor informa si el acceso ha sido denegado (si no ha detectado la etiqueta UID) o si ha sido aceptado, mostrando el nombre de usuario asociado a la etiqueta UID de la tarjeta o llavero.

3. INDICADOR SONORO CON BUZZER PASIVO

3.1. Montaje

Para el montaje del aviso sonoro no es necesario modificar nada del circuito de la tarea anterior (ver Punto 2.1). El avisador sonoro elegido es un buzzer pasivo genérico de 3 pines, masa (negro), alimentación (rojo) y señal (amarillo).

Para la señal se ha elegido la salida digital 7 del Arduino.

El esquema del circuito será el siguiente:

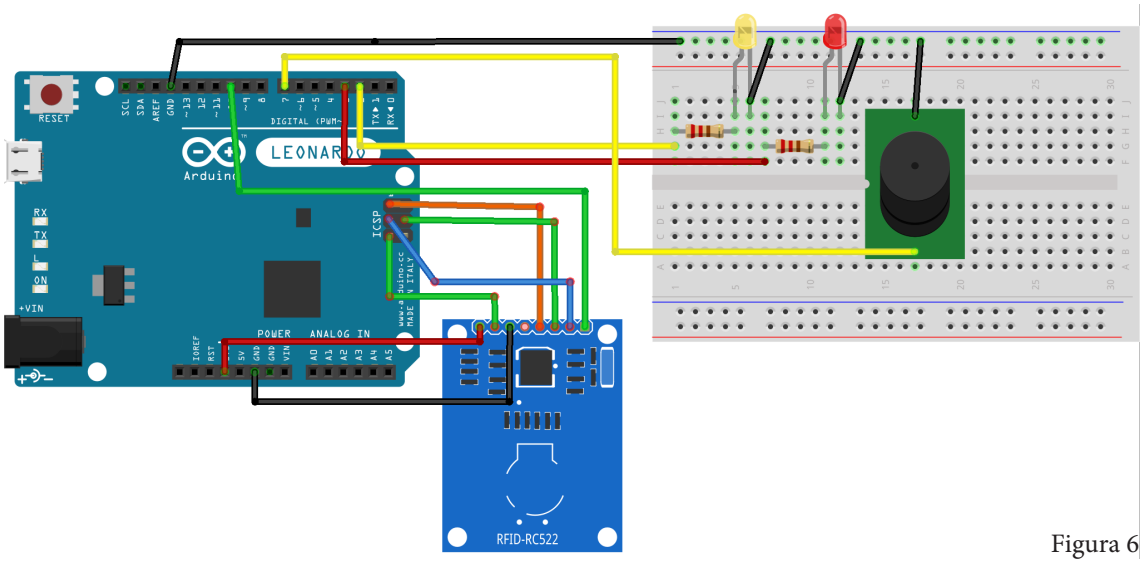


Figura 6

En el esquema no se muestra el cable de alimentación del buzzer (rojo) debido a la limitación de componentes del programa de edición.

3.2. Código

A la hora de programar el buzzer se ha especificado el pin que llevará la señal al buzzer como: `pinMode(buzzerPin,OUTPUT);` habiendo especificado previamente que la variable `buzzerPin` es igual a 7.

A la hora de especificar que se quiere enviar una señal al buzzer, se utiliza la siguiente expresión: `tone(7,300,100);` donde el `7` indica el pin digital de salida, el `300` la frecuencia del sonido (Hz) y el `100` la duración del sonido en milisegundos.

Con la expresión `noTone(7);` se detiene cualquier sonido en curso parando el envío de señal a través del pin 7.

El código completo del Trabajo se adjunta en la carpeta del proyecto.

3.3. Pruebas

Cuando un usuario se identifica correctamente, el buzzer emite un sonido agudo continuo, en cambio cuando una identificación ha sido denegada, el buzzer emite un sonido algo más grave y discontinuo. El sistema de aviso sonoro funciona según lo estipulado. Se ha cumplido el objetivo de la siguiente tarea:

3. Añadir al sistema un indicador sonoro con buzzer pasivo

4. PANTALLA LCD 16x2

4.1. Montaje

La pantalla elegida es un display de cristal líquido que dispone de 2 líneas de texto con 16 caracteres en cada una. Dispone de 16 pines para conectar al Arduino, pero también existe un dispositivo externo que facilita enormemente su conexión.

Este dispositivo, llamado I²C (también escrito como I2C), es un bus de comunicaciones Serie que utiliza dos líneas para transmitir la información: una para los datos y otra para la señal de reloj. En la mayoría de las placas Arduino, SDA (línea de datos) está en el pin analógico 4, y SCL (línea de reloj) está en el pin analógico 5. En el caso del Arduino Leonardo que nos ocupa, el pin SDA es el 2 digital y SCL es el 3 digital. También hay que hacer notar que su alimentación es de 5V, no de 3.3V.

La conexión, por tanto, es tremendamente sencilla, teniendo que conectar tan solo 4 cables. El problema radica en que anteriormente los pines digitales 2 y 3 estaban usados por los LEDs amarillo y rojo del sistema de aviso luminoso. Así que la conexión para esta tarea se modifica para ubicar las salidas de los LED's en los pines digitales 4 y 5 respectivamente.

El esquema del circuito se muestra a continuación:

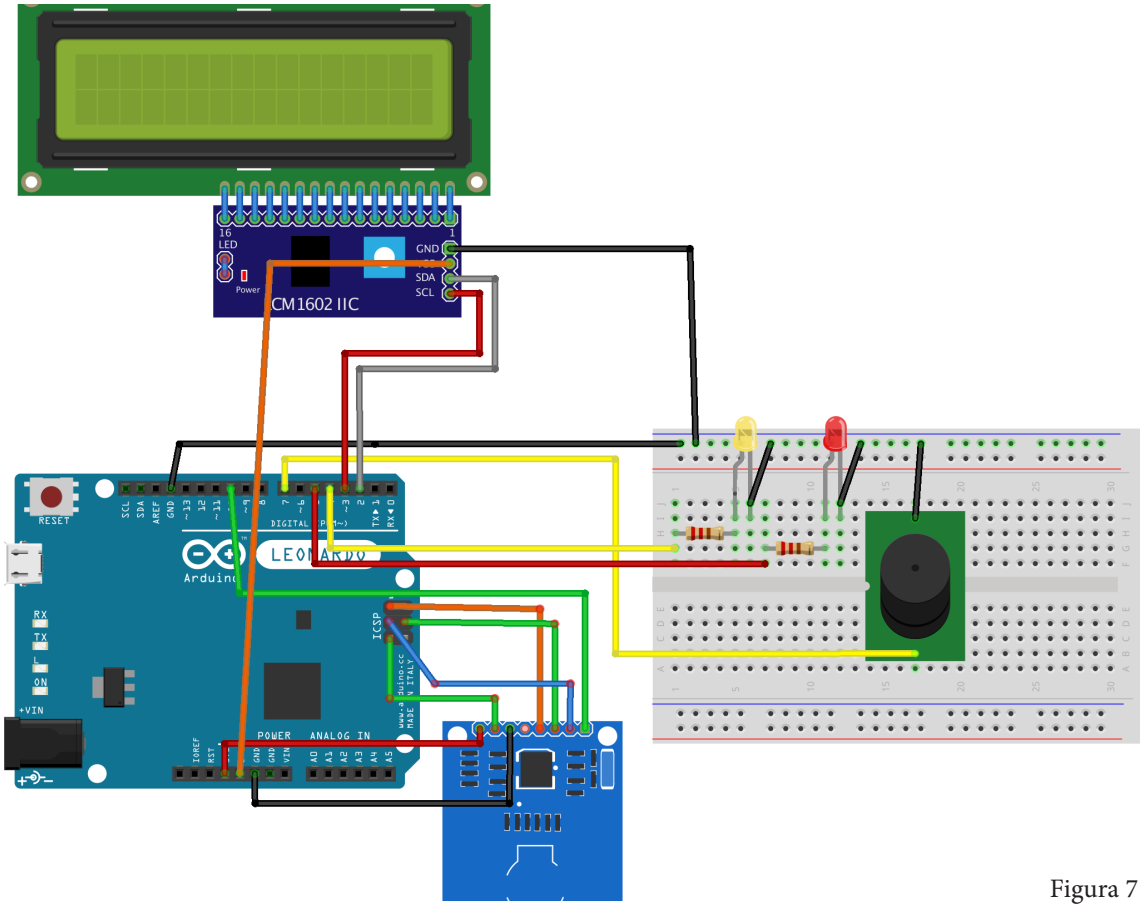


Figura 7

4.2. Código

La dificultad de las pantallas LCD genéricas es la diversidad tan grande que hay entre productos de multitud de fabricantes. Y hay que tener en cuenta además, que el dispositivo I2C dispone también de multitud de librerías diferentes. Todo esto hace complicado que la pantalla funcione a la primera.

Una vez hallada la librería correcta, la utilización de la pantalla no es complicada para usos normales. En este caso se necesita mostrar información por pantalla, prácticamente la misma información que en etapas anteriores se mostraba en el Monitor del IDE de Arduino. Por tanto se necesita un aviso inicial que inste al usuario a acercar su tarjeta/llavero NFC y posteriormente un feedback visual que indique si la identificación ha sido correcta o no.

Básicamente, y siguiendo la estructura del código generado en etapas anteriores, se necesita escribir el aviso inicial por pantalla y cuando el usuario se identifique correctamente que se borre la pantalla y que aparezca el aviso de detección correcta. Lo mismo ocurre con la detección fallida, primero debe borrarse la pantalla. Además, una vez mostrado el aviso de detección correcta/fallida, en la pantalla debe aparecer de nuevo el aviso inicial.

Para ello se ha creado la función `void reinicioLCD()`, que será llamada en 3 partes diferentes del código y por tanto supone un ahorro de líneas de código (10 concretamente). Esta función escribe el texto inicial de vuelta en la pantalla.

Otra función de la librería de la pantalla LCD, y que además será la más utilizada, es: `lcd.setCursor(0,1);` Esta permite indicar al sistema en qué posición de la pantalla debe empezar a escribir caracteres. El primer número indica la posición del caracter (el 0 es la posición más a la izquierda) y el segundo número indica la línea (0 la de arriba y 1 la de abajo).

El código completo del Trabajo se adjunta en la carpeta del proyecto.

4.3. Pruebas

Los testeos, al igual que en las dos etapas anteriores, son sencillos de comprobar, ya que no hay almacenamiento de datos de ningún tipo y todos los resultados son correctos de cara a los objetivos programados.

Por tanto, llegado a este punto, se ha cumplido el objetivo de la siguiente tarea:

4. Añadir al sistema una pantalla LCD para indicar a la persona que desea acceder si el acceso o la salida se ha realizado correctamente.

5. TECLADO NUMÉRICO

5.1. Montaje

El teclado utilizado para esta funcionalidad es un modelo genérico con matriz de 4x4 teclas (10 teclas numéricas, asterisco, almohadilla y las letras A, B, C y D). Su funcionamiento es el siguiente: cuando una tecla es pulsada, la señal de su columna y fila es enviada al Arduino, que la interpretará según se configure en el IDE. Por ejemplo, la tecla 1 estará posicionada en la columna 1 y fila 1, mientras que la tecla 8 estará posicionada en columna 2 y fila 3. Habrá, por tanto, 8 cables que conectarán el teclado con el Arduino.

La conexión se realiza en los INPUTS 2 a 9 del Arduino, tal como muestra el siguiente esquema:

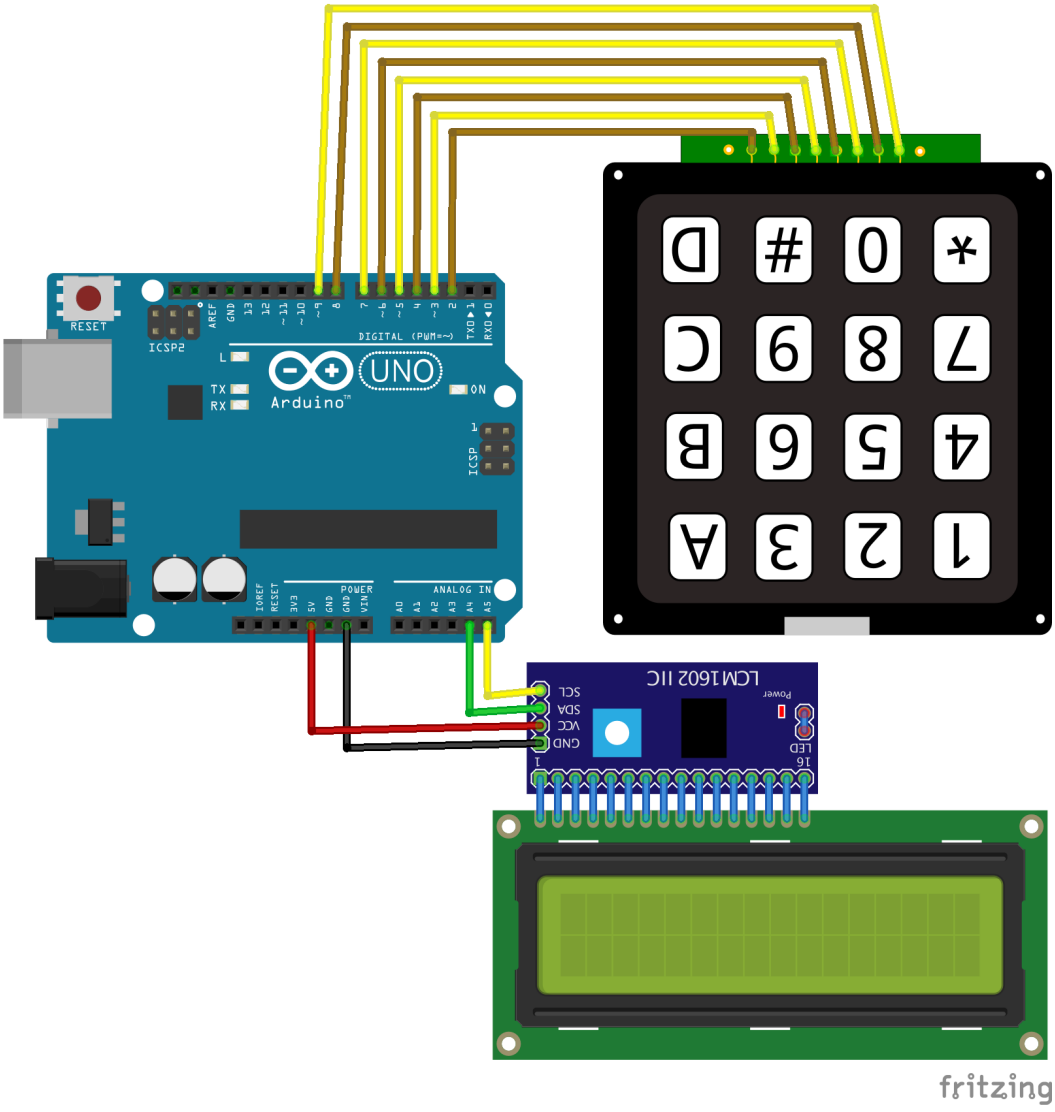


Figura 8

5.2. Código

El código de esta implementación es algo más complejo que en las anteriores, por lo que se desarrollará con más detalle.

Es necesario añadir la librería adicional `Keypad.h`, y la forma de crear la asociación de las teclas (como combinación de una fila y una columna) con los pines del Arduino se realiza de la siguiente manera:

```
const byte Filas = 4;     //Cuatro filas
const byte Cols = 4;    //Cuatro columnas

byte Pins_Filas[] = {9, 8, 7, 6};
byte Pins_Cols[] = { 5, 4, 3, 2};

char Teclas [ Filas ][ Cols ] =
{
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
```

La primera parte del código indica el tamaño de las filas y columnas, en este caso 4 cada una. La segunda parte del código asocia a una variable array de tipo byte los pines a los que se conecta cada cable. Por ejemplo, las filas se conectan a los pines 6, 7, 8 y 9. La tercera parte del código es la matriz de asociaciones que finaliza la asociación entre cada tecla y lo que interpretará el Arduino cuando se pulse esa tecla. En este caso se ha representado tal cual es físicamente el teclado numérico, y se ha guardado en la variable llamada `Teclas`.

A continuación, se genera la variable `Teclado1` de tipo `Keypad`, incluída en la librería asociada al inicio del código, incluyendo las variables guardadas antes:

```
Keypad Teclado1 = Keypad(makeKeymap(Teclas), Pins_Filas, Pins_Cols, Filas, Cols);
```

Para guardar la tecla pulsada en cualquier momento, se debe utilizar la función `getKey()` de la librería `Keypad.h`, y se almacena en una variable tipo char:

```
char key = Teclado1.getKey();
```

Una vez almacenada la tecla pulsada, ya puede ser analizada. El problema radica en que el PIN solicitado es de 4 dígitos, por tanto el sistema debe almacenar 4 pulsaciones seguidas del teclado.

Para ello se convierte la variable `char key` en una cadena de texto llamada `stringOne`, que a continuación se concatenará en una nueva cadena llamada `cadenaPIN`. Cada vez que una tecla es pulsada, su valor se convierte en una cadena y finaliza concatenándose en la cadena final `cadenaPIN`:

```
String cadenaUno =  String(key);
cadenaPIN =  cadenaPIN + cadenaUno;
```

Cuando `cadenaPIN` es de longitud 4 (cuando se han pulsado 4 teclas) y los 4 caracteres en esa cadena coinciden con los caracteres del PIN de algún empleado (guardados en variables tipo `String pinvalido="1346";`), el sistema avisa mediante la pantalla LCD que el acceso se ha permitido correctamente.

Una vez que el sistema ha permitido el acceso o lo ha denegado, el contador debe ponerse a cero (estará en 4) y ambas cadenas deben vaciarse para poder volver a almacenar los 4 caracteres del PIN:

```
contadorpin=0;
cadenaPIN.remove(0);
cadenaUno.remove(0);
```

5.3. Pruebas

Para la comprobación del correcto funcionamiento del teclado, se estipula como válido para el usuario Miguel el PIN '`1346`'. Nada más arrancar el sistema, la pantalla LCD indica al usuario que introduzca el PIN.

Si el PIN es erróneo, en la pantalla se mostrará: `MAL! ACCESO DENEGADO`

Si el PIN es corecto, en la pantalla se mostrará: `BIEN! ADELANTE`

Se ha cumplido el objetivo de la siguiente tarea:

5. Añadir al sistema un teclado numérico para permitir a la persona que quiere acceder introducir un pin en vez de usar el contacto NFC.

6. ETHERNET Y BASE DE DATOS

6.1. Montaje

Llegado este punto, el Arduino Leonardo usado hasta ahora se muestra insuficiente para continuar, debido a las entradas que presenta y a la posición de sus puertos SCL, SDA, etc.

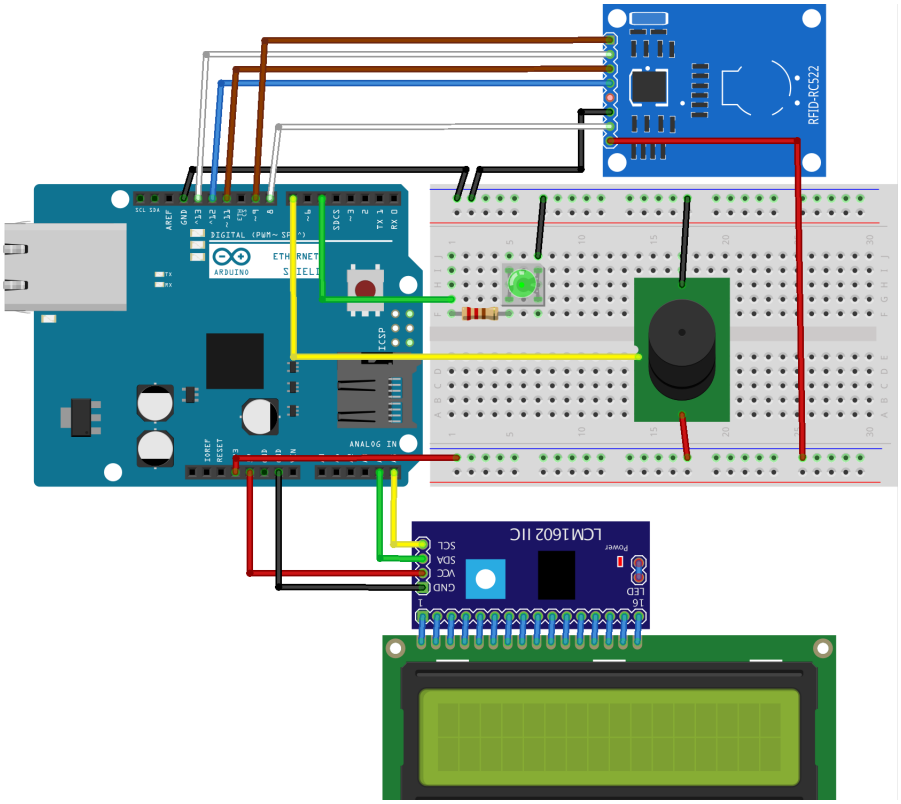
Aunque en el apartado 5 anterior este problema no era importante, ya que el teclado se puede conectar sin problemas, ya se decidió cambiar el montaje, los esquemas y el código para trabajar desde un Arduino Uno en vez del Leonardo.

A la hora de trabajar con el Shield Ethernet, la conexión es sencilla: tan solo debe ir situado encima del propio Arduino, haciendo coincidir todos los pines (incluso los del socket de 6 pines). El Shield usará los pines 8, 9, 10, 11, 12 y 13 para comunicarse con el Arduino. Se han de conectar los pines RST y SDA del lector en los pines 8 y 9 del Shield Ethernet. Se verá su configuración más adelante.

Si se mantienen conectados en el 9 y el 10, el lector RFID y el Shield Ethernet no funcionarán. Esto se comprueba fácilmente conectando y desconectando el lector RFID y comprobando entonces la conexión de red. También se ha decidido eliminar uno de los LEDs de aviso y cambiar el restante por uno de alta potencia y color verde. Esto hará que solo se ilumine el LED en el caso de una detección correcta de tarjeta o llavero y que sea más notorio debido a su mayor luminosidad.

El esquema de la nueva conexión se muestra a continuación:

Figura 9



6.2. Código

Se introduce en el código al librería `Ethernet.h` y se define un nuevo cliente con la línea de código: `EthernetClient client;`

Como se comentaba anteriormente, si se mantiene la conexión del lector RFID en los pines de entrada 9 y 10, tanto el lector RFID como el Shield Ethernet no funcionarán. Para ello, el lector RFID se conecta a los pines 8 y 9 y se escribe el siguiente código:

```
pinMode(9, OUTPUT);
digitalWrite(9, HIGH);
pinMode(10, OUTPUT);
digitalWrite(10, LOW);
```

Con ello se consigue que el lector RFID siga realizando su función y que el Shield Ethernet funcione correctamente.

A partir de este momento, se considera que el router al que se conecta el sistema dispone de un servidor DHCP y que este está activado. Así, al encender el Arduino y conectarlo al router, este le asignará una dirección IP libre sin necesidad de configuración. Por supuesto, se podrían asignar direcciones fijas tanto de IP, como de Puerta de Enlace y Máscara de Subred. Para conocer si el servidor DHCP no funciona o si el Arduino está conectado erróneamente, se utiliza:

```
if (Ethernet.begin(mac) == 0)
{
   Serial.println("Fallo usando DHCP");
   for (;;)   ; //No continúa
}
```

Ya dentro del código del `loop()`, se podrán utilizar las funciones de la librería `Ethernet.h` para mostrar información de la conexión. En este caso, una vez detectado correctamente un acceso (tarjeta o llavero), el Monitor Serial mostrará la dirección IP del dispositivo Arduino. Para ello se utiliza la función `Ethernet.localIP()`:

```
Serial.println("\nBIENVENIDO A LA OFICINA, MIGUEL");
Serial.print("La dirección IP del Arduino es: ");
Serial.println(Ethernet.localIP()); //Muestra la IP en el Monitor Serial
Serial.println();
```

La pantalla LCD seguirá mostrando el mensaje de bienvenida o el mensaje de error al usuario que está intentando acceder.

A continuación se desarrolla el sistema que enviará el nombre de la persona que está intentando identificarse y que lo introducirá en una base de datos MySQL[7].

[7] *https://www.mysql.com/*

No es necesaria ninguna librería adicional, ya que las funciones a utilizar están dentro de la librería `Ethernet.h` utilizada anteriormente.

Se trabajará con una base de datos MySQL (de ahora en adelante BDD) hospedada en un servidor dedicado, por tanto dispondrá de una dirección IP fija. Para dirigir el envío de los datos al servidor, se especifica su IP:

```
byte server[] = { 185,68,108,62 }; // ip del Hosting
```

Dentro del bucle que decide si el UID de la tarjeta está permitido o no por el sistema (el bucle `IF` principal del programa), se escribirán dos envíos de datos, uno dentro del `IF` y otro dentro del `ELSE`.

Dentro del `IF` (se entra al bucle si se ha reconocido el UID de una tarjeta o llavero con autorización) se conecta al servidor como un cliente utilizando la función:

```
if (client.connect(server, 80))
```

Una vez conectado al servidor, se escriben las siguientes instrucciones:

```
client.print("GET http://goblintrader.es/miguel/arduino.php?valor=");
client.print("Miguel");
client.println(" HTTP/1.0");
client.stop();
Serial.println("Datos enviados correctamente a la BDD.");
```

Las primeras 3 líneas envían al servidor una petición `GET`[8] con una variable llamada `valor` y el nombre de la persona que se ha conectado correctamente (en este caso, '`Miguel`').

La cuarta línea cierra la conexión al servidor, de lo contrario al leer una nueva tarjeta o llavero, el Arduino indicará que no se ha podido establecer la conexión con el servidor.

La última línea mostrará en el Monitor de Arduino que se ha conectado correctamente al servidor y que los datos han sido enviados.

Dentro del `ELSE` (se entra al bucle si NO se ha reconocido el UID de una tarjeta o llavero con autorización) se ejecutan las mismas instrucciones que dentro del `IF` explicadas anteriormente, pero en vez de enviar el nombre de la persona que ha recibido el acceso, se envía el texto 'Error'.

Por tanto, si una persona se registra correctamente, su nombre aparecerá en la BDD, pero si se ha producido un intento de acceso no autorizado, será el texto 'Error' el que aparecerá en la BDD.

[8] *http://php.net/manual/es/reserved.variables.get.php*

Para poder leer el dato enviado se crea un archivo PHP[9] que recibirá el dato y lo enviará a la BDD. El archivo se compone de 3 instrucciones de código:

```
$conexion = mysql_connect("localhost", "XXXXX", "YYYYY");
mysql_select_db("ZZZZZ",$conexion);
mysql_query("INSERT INTO `zzarduino`(`nombre`) VALUES ('" . $_GET['valor'] . "')", $conexion);
```

La primera instrucción genera una conexión con la BDD, donde XXXXX es el usuario de la base de datos e YYYYY es la contraseña. La segunda instrucción conecta directamente a la BDD, llamada ZZZZZ.

Finalmente, la tercera instrucción utiliza una función del lenguaje SQL[10], `INSERT INTO`, para escribir en la tabla denominada 'zzarduino' el dato de la variable `valor` enviado por el Arduino.

La BDD dispondrá de una tabla creada especialmente para los registros del sistema. La tabla se llamará 'zzarduino' y dispondrá de 3 columnas:

- id: número autocorrelativo único para registrar cada operación.
- fecha: debido a que Arduino no dispone de reloj interno (aunque se podría instalar un módulo para conseguirlo), este campo guarda la fecha en la que se recibe la petición de `INSERT` en la base de datos.
- nombre: el valor del texto recibido desde el archivo PHP. El texto será el nombre de un usuario si un acceso ha sido validado, o el texto 'Error' si el acceso ha sido no autorizado.

El archivo PHP utilizado en este punto se incluye en la carpeta llamada 6_Ethernet adjunta a este trabajo.

6.3. Pruebas

En las pruebas del sistema se han realizado múltiples accesos, tanto válidos como erróneos. Tanto el contador autocorrelativo como la fecha funcionan correctamente y se muestran en los registros de la base de datos.

El Monitor Serial del Arduino muestra la información del envío al archivo PHP correctamente, y gracias a ello se ha podido identificar uno de los principales problemas: cuando el sistema leía una tarjeta correcta y enviaba los datos correctamente al archivo PHP, no se permitía volver a leer una nueva tarjeta válida. O mejor dicho, sí se podía leer la tarjeta, pero los datos no se enviaban al archivo PHP y por tanto no se escribían en la BDD. Para ello se escribieron las líneas de texto que informaban si la conexión se había realizado correctamente o no.

[9] *Pre Hypertext -processor*
[10] *Structured Query Language*

Gracias a estas instrucciones de ayuda, se observó que una vez leída una tarjeta correcta, el servidor no se desconectaba y se mantenía la sesión abierta. Ello producía que al intentar registrar una nueva tarjeta el sistema indicara que la conexión al servidor no se había llegado a realizar.

La solución fue escribir la función de cierre de conexión con el servidor que proporciona la librería `Ethernet.h`, tanto al finalizar el `IF` como al finalizar el `ELSE`:

```
client.stop();
```

Llegado este punto, los valores de acceso registrados en la BDD solo pueden ser analizados en la propia base de datos usando programas tipo *phpMyAdmin*[11] o gestores de bases de datos de escritorio.

En el Punto 8 (ver página 35) de este mismo trabajo se creará una aplicación web basada en HTML[12] y PHP para poder visualizar de una manera más cómoda los registros de la BDD. Se podrán seleccionar usuarios, filtrarlos por nombre y también por fecha, seleccionando rangos.

Se ha cumplido el objetivo de la siguiente tarea:

6. Añadir al sistema una tarjeta Ethernet para poder comunicar el Arduino con Internet y configurar el sistema para que guarde los datos de acceso en una base de datos.

[11] *https://www.phpmyadmin.net/*
[12] *HyperText Markup Language*

7. TARJETA WIFI Y MONTAJE AVANZADO

7.1. Montaje

El módulo wifi[13] utilizado en este punto está basado en el chip ESP8266, una alternativa barata y de tamaño muy reducido a los Shields wifi de Arduino. El módulo dispone de 8 pines para su conexión, aunque se utilizarán tan solo 5: TX, RX, GND, CH_PD y VCC.

El módulo se alimenta mediante su pin VCC a 3.3V, y tiene una tolerancia más bien pequeña; se ha comprobado en este apartado que tensiones inferiores a 3.3V suponen comportamientos extraños del dispositivo y que tensiones superiores a 4.5V pueden quemarlo. El pin CH_PD se alimenta también en los 3.3V.

Aunque según su hoja técnica, los pines TX y RX deben conectarse a 3.3V (o a 5V con un divisor de tensión), no ha habido problemas conectándolos directamente al Arduino (a los pines 2 y 3). El esquema de conexión es el siguiente:

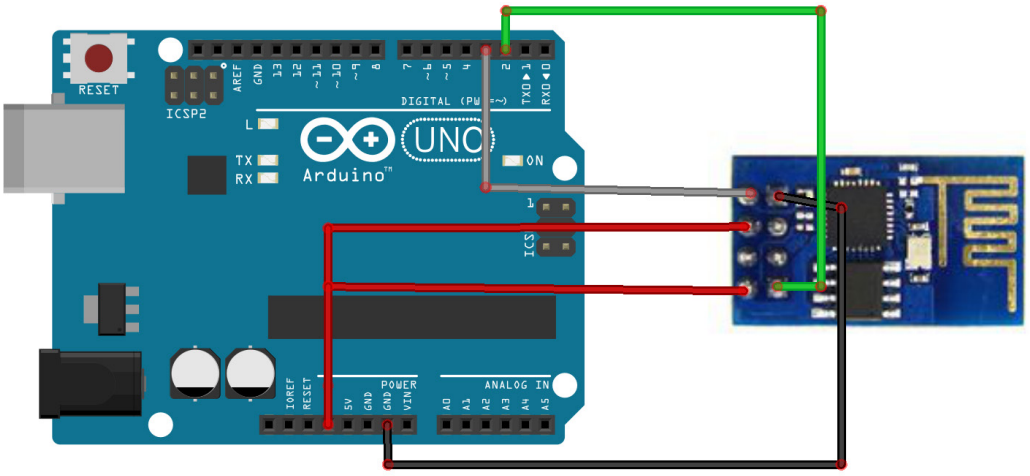


Figura 10

Como se expondrá más adelante, el Arduino Uno utilizado hasta ahora es insuficiente para poder realizar la funcionalidad necesaria correctamente.

Siendo así, a partir de este momento se comienza a utilizar un Arduino Mega 2560 que permitirá tener mejores opciones de comunicación Serial, tanto en cantidad de puertos como en velocidad de transisión en baudios.

Debido al cambio de puertos del nuevo Arduino, la conexión varía con respecto al Arduino Uno, así que el nuevo esquema de conexión es:

[13] *El término wifi, sustantivo común escrito normalmente en redonda (sin comillas ni cursiva), proviene de la marca comercial Wi-Fi. Es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.*

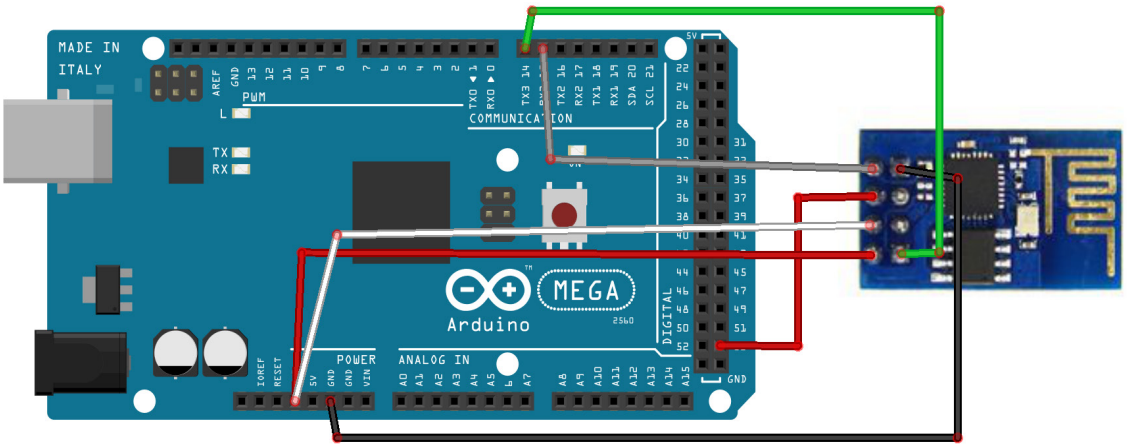


Figura 11

Se realizan algunos cambios en las conexiones con respecto al Arduino Uno:

- El puerto RST de reset se conecta también a 3.3V.
- El puerto CH_PD se conecta al puerto 53. Este puerto digital se configurará en el código como siempre activo para que el puerto siempre esté activo.
- Los puertos TX y RX se conectan a los puertos RX3 y TX3, que son los puertos de la comunicación Serial 3.

7.2. Código y Pruebas

Debido a la cantidad de código que requieren los apartados anteriores, y a la dificultad de este apartado, se decide realizar un código limpio partiendo desde cero. Se escriben las líneas necesarias para poder conectar el dispositivo a una red wifi.

El dispositivo ESP8266 se comunica con el Arduino mediante comunicación Serial. Arduino dispone de una librería para ello, instalado con el IDE de fábrica, llamada `SoftwareSerial.h`. Será utilizada en el código de este apartado como la única librería.

Se comienza especificando los puertos de transmisión para la comunicación Serial por Software (la que genera la librería `SoftwareSerial.h`). Se conecta a los puertos 2 y 3 del Arduino:

```
SoftwareSerial ESP8266(3, 2); // RX | TX
```

Se inician ambos Seriales, el del Monitor Serial habitual y el nuevo, con una velocidad de 9600 baudios:

```
Serial.begin(115200);
BT1.begin(115200);
```

En el `void loop()` se especifican las transmisiones y recepciones de datos entre ambos Seriales:

```
void loop()
{   String B= "." ;
     if (ESP8266.available())
         { char c = ESP8266.read() ;
           Serial.print(c);
         }
     if (Serial.available())
         {  char c = Serial.read();
            ESP8266.print(c);
         }
}
```

Una vez cargado el código en el Arduino comienzan los problemas de comunicación. La instrucción más sencilla para comunicarse con el ESP8266 es el comando `AT`, que debe devolver un simple `OK` para comprobar que la comunicación está abierta. Pero nada ocurre en el Monitor ni hay ninguna respuesta.

Se revisa la hoja de especificaciones del ESP8266 para comprobar que su velocidad de trabajo es de 115200 baudios, así que al haber abierto las comunicaciones Seriales a 9600 baudios, nada se comunica entre ambos dispositivos. Se cambian ambas velocidades a 115200 y al iniciar el sistema ya se observa texto, pero son caracteres ininteligibles y lo que parece ser es la marca fabricante del dispositivo:

```
[illegible]sll[illegible]s [illegible]c [illegible]o  [illegible] [illegible]l[illegible] p[illegible]l[illegible][illegible]p [illegible][illegible]l cl c[illegible]p?[illegible]o[illegible]loo[illegible] c c p[illegible][illegible]cl` slp[illegible]o[illegible][illegible] [illegible][illegible][illegible]l
Ai-Thhnker Technonogy Co. Ltd/
```

Comprobando con la instrucción `AT`, el sistema responde bien con un `OK`. La siguiente instrucción es `AT+CWLAP`, que muestra un listado de las redes wifi detectadas por el dispositivo ESP8266. En este caso se encuentran varias, pero el problema de la codificación de caracteres sigue afectando a la visualización. En la siguiente captura se muestran 5 instrucciones AT+CWLAP seguidas:

```
OK AT+CWLAP
+CWLAP:(3,"pdpephone_ADSLOL6N",-66,#5a:42:c6:d4;65:f0",1,5,0)
+CWA":A":+A4+E6,"f,:"2)7f)s0+R:+R6)b][illegible]  " b
g50A4,g9,":1Od,AT+CWLAP
+CWL@P:(3,"pepephone_ADSLOL6O",-67,"5a:43:c6:d4:65:g0",1,5,0)
+CW3-:W60PD9P84PFc)gf
s0W7aCe:T4,g0d1,0:,",1:-0P,eW0a+1c
":OAT+CWLAP
+CWLAP:(4,"Orange-A070",-73,"4c:09:d4:87:0e:75",1,0,0)
+CWLAP:(3,A:I92I41",1:,:P,7Ad,",fA"2Wn0ef+Se-(-0P,:WF5)e:)-80le)a93AT+CWLAP+CWI
+CWLAP:(4,"JAZ,f(OdLA:)A18":,,,:A"aL":C5:C-:0n69a:,355A55)e61zc",,844d("1""1AT-
+CWLAP:(3,#TheHaus",-74,"d4:7b:b0:g0:29:40",1,-9,0)
+CWL@P:(3,"MO":C261p76LF:L0e:"5:"6:973-d4,,27:A8dWR60nd)S:9a0,"e1(,:A"[illegible]"b
te)A60AT+CWLAP
+CWLAP:(3,"pepephone_ADSLOL6O",-67,"5a:43:c6:d4:65:f0",1,5,0)
+CW(76
```

Otro problema es que cada vez que se ejecuta `AT+CWLAP`, las redes encontradas son diferentes. El sistema no es capaz de mostrar todas las redes wifi encontradas a la vez.

Una búsqueda de información en Internet deja ver que el Arduino Uno utilizado no soporta bien la comunicación Serial con valores altos de baudios. Así que no es posible utilizar cómodamente el dispositivo ESP8266 con un Arduino Uno. Incómodamente sí es posible utilizarlo, ya que responde a las instrucciones `AT`, pero la visualización de los caracteres y los futuros problemas que pueden surgir a raiz de la diferencia de velocidades de transmisión hace valer la opción de mejorar el modelo de Arduino por uno superior.

Otro problema es que se están utilizando los puertos digitales 2 y 3 del Arduino, que no están pensados para la comunicación Serial con otros dispositivos, sino para recibir o enviar pulsos digitales. Sí podrían usarse los puertos digitales 0 y 1 del Arduino Uno, que son los puertos de transmisión y recepción Serial TX y RX, pero los errores al estar conectados los cables a esos puertos son continuos (lo ideal es que estos puertos se dejen para la comunicación Serial interna con el Monitor Serial).

Se decide por tanto adquirir un Arduino Mega, cuya comunicación Serial no tiene problema para trabajar a 115200 baudios y que dispone de 3 comunicaciones Serial hardware adicionales. El esquema de conexión del nuevo dispositivo se muestra en la Figura 11. Al cargar el código y abrir el Monitor Serial ya se pueden ver textos sin caracteres extraños, y al utilizar el comando `AT+CWLAP`, se observan correctamente todas las redes wifi detectadas por el ESP8266:

```
AT+CWLAP
+CWLAP:(4,"Orange-A060",-45,"4c:09:d4:87:0e:75",1,0,0)
+CWLAP:(3,"MOVISTAR_6A28",-67,"e2:41:36:0b:6a:28",1,-12,0)
+CWLAP:(2,"MOVISTAR_46B8",-80,"f8:ed:80:20:46:c1",1,0,0)
+CWLAP:(0,"Chromecast6725.b",-86,"fa:8f:ca:6c:3f:8d",1,-2,0)
+CWLAP:(4,"Orange-EF3A",-89,"88:03:55:e2:ef:3c",1,-7,0)
+CWLAP:(3,"pepephone_ADSLOL6N",-85,"5a:42:c6:d4:65:f0",1,5,0)
+CWLAP:(2,"MOVISTAR_EDF4",-92,"f8:8e:85:fa:ed:f5",1,-7,0)
+CWLAP:(3,"MOVISTAR_A9FC",-87,"fc:b4:e6:dc:a9:fd",1,0,0)
+CWLAP:(3,"MOVISTAR_914D",-56,"98:97:d1:34:91:4e",1,30,0)
+CWLAP:(2,"MOVISTAR_7516",-87,"f8:8e:85:2a:75:17",1,-4,0)
+CWLAP:(4,"Orange-7245",-92,"9c:80:df:4d:72:47",1,-2,0)
+CWLAP:(3,"TheHaus",-85,"d4:7b:b0:f0:29:40",1,-9,0)
+CWLAP:(4,"MASMOVIL_k3XK",-82,"00:4a:77:61:44:b2",2,-4,0)
+CWLAP:(3,"UESD-PERSONAL",-95,"f0:b0:52:32:a4:89",5,-7,0)
+CWLAP:(4,"JAZZTEL_E7F8",-91,"d8:b6:b7:7c:e7:f8",1,-7,0)
+CWLAP:(4,"Orange-BDF2",-77,"9c:80:df:72:bd:f4",6,-2,0)
+CWLAP:(4,"Orange-E75F",-69,"d0:05:2a:55:e7:61",8,11,0)
+CWLAP:(3,"vodafone1880",-77,"e0:60:66:b9:18:81",9,-12,0)
+CWLAP:(4,"JAZZTEL_HOGAR ",-88,"00:b6:b7:0e:90:94",9,-12,0)
+CWLAP:(4,"Jazztel_20",-80,"4c:ed:de:fa:3d:01",11,8,0)
+CWLAP:(3,"MOVISTAR_F5A8",-88,"e2:41:36:00:f5:a8",11,-22,0)
+CWLAP:(1,"WLAN_77H",-91,"dc:0b:1a:70:7c:ba",11,-2,0)
+CWLAP:(3,"MOVISTAR_3B59",-89,"fc:b4:e6:e9:3b:5a",11,-9,0)
```

Para conectarse a un red se utiliza el comando `AT+CWJAP="ID","contraseña"`:

```
AT+CWJAP="MOVISTAR_914D","[redacted]"
WIFI DISCONNECT
WIFI CONNECTED
WIFI GOT IP

OK
```

La conexión se realiza correctamente y es estable. Para comprobar si al dispotivo se le ha asociado una dirección IP, se utiliza el comando `AT+CIFSR`:

Se observa que la dirección IP del ESP8266 es 192.168.1.44

```
AT+CIFSR
+CIFSR:APIP,"192.168.4.1"
+CIFSR:APMAC,"5e:cf:7f:90:70:86"
+CIFSR:STAIP,"192.168.1.44"
+CIFSR:STAMAC,"5c:cf:7f:90:70:86"

OK
```

Una vez establecida la conexión con la red y conectado el sistema a Internet, el próximo paso es enviar un dato al archivo PHP `arduino.php`. Para ello se realizará una petición `GET` enviando un dato de texto. El propio archivo PHP será el encargado de enviar el datos de texto a la base de datos MySQL. Todo esto se explicó en el punto anterior (ver punto 6.2 en la página 23).

En este caso, en vez de poder enviar los comandos desde propio código de Arduino en el IDE, se deben enviar los comandos mediante el Serial de la misma manera, utilizando los comandos `AT`.

Para enviar un dato se necesitan realizar 3 pasos:

- Abrir una conexión con el servidor de la manera: `AT+CIPSTART="TCP","goblintrader.es",80`. No es necesario especificar `http://` ni `www.` y es válido indicarlo como IP en vez de como dominio.
- Especificar el número de bytes que se enviarán con el GET: `AT+CIPSEND=72`. Si se especifican menos bytes que el propio mensaje a enviar, este será cortado donde termine el número de bytes especificado en este comando.
- Enviar el propio dato, especificando la localización el archivo PHP que recibirá el GET: `GET/arduino.php?valor=hola HTTP/1.1 Host=goblintrader.es\r\n\r\n`

Realizados estos 3 pasos en el Monitor Serial, el resultado es el siguiente:

```
AT+CIPSTART="TCP","goblintrader.es",80
CONNECT

OK
AT+CIPSEND=84

OK
> CLOSED
```

El primer paso se realiza sin problemas, el sistema se conecta al servidor. El segundo paso produce una parada en el sistema y a los pocos segundos el mensaje `CLOSED` aparece en pantalla, indicando que se ha cerrado la conexión con el servidor.

Esto evita que se pueda introducir el comando que envía el dato al PHP.

Después de varias comprobaciones, se detecta que es problema del servidor; si solo se introduce el primer comando de conexión, se conecta correctamente pero se desconecta a los 10 segundos. Probado con la conexión a google.es, el sistema no se desconecta y la conexión se mantiene por lo menos algunos minutos.

Por supuesto, al no disponer del archivo arduino.php en los servidores de Google, el segundo paso se puede ejecutar pero el tercero no.

Después de corregir el acceso en el servidor (cambiando de servidor a modelbrush.com, el sistema permite finalizar las 3 instrucciones:

```
OK
AT+CIPSEND=72
OK
>
busy s...

Recw 72 bytes

SEND OK
CLOSED
```

El Serial Monitor indica que la conexión con el servidor se ha realizado `OK`, informa también en forma de `OK` que espera recibir un mensaje de 72 bytes y al enviar la petición `GET` (el tercer comando), también lo valida con un `OK`. Pero también aparece un `busy` (ocupado en castellano), que indica que se han enviado más bytes de los especificados, y por tanto ningún dato será enviado.

Y aquí es donde se complica aún más la cosa.

Para enviar mediante comando `AT` en la consola un comando, hay que eliminar los `\r\n`, que son los saltos de línea y retornos de carro. Esto es así porque la propia consola añade estos caracteres en cada retorno de carro. Y también hay que tener en cuenta que cada `\r` y `\n` contarán solo como 1 caracter.

Por ejemplo: `GET \arduino.php?valor=hola HTTP/1.1\r\n` serán 38 caracteres.

También se debe separar el último comando en varias líneas para que la petición `GET` funcione correctamente. Y hay que darse cuenta que la petición termina con un doble `\r\n\r\n`, que serán 4 caracteres.

Aparte, la petición global DEBE finalizar con un `\r` para indicar que la cadena de caracteres ha sido enviada totalmente. Por tanto, sumando los `\r\n` del final de la línea y otro `\r` necesario para finalizar, se necesitan añadir 3 caracteres más a la suma total. Como el comando es de 60 caracteres, se queda finalmente en 63.

Por tanto los envíos de texto en la consola serán:

- `AT+CIPSTART="TCP","modelbrush.com",80`
- `AT+CIPSEND=63`
- `GET \arduino.php?valor=hola HTTP/1.1` (38 caracteres)
- `Host=modelbrush.com` (22 caracteres)
- Retorno de Carro (2 caracteres)
- Retorno de Carro (1 caracter)

Y el resultado en la consola del Monitor Serial:

```
------
AT+CIPSTART="TCP","modelbrush.com",80
CONNECT

OK
AT+CIPSEND=63

OK
>
busy s...

Recv 63 bytes

SEND OK

+IPD,159:HTTP/1.1 200 OK
Date: Fri, 13 Jan 2017 21:24:31 GMT
Server: Apache
X-Powered-By: PleskLin
Content-Length: 0
Connection: close
Content-Type: text/html
```

Si se comprueba la tabla `zzarduino` de la base de datos se puede ver que los textos han llegado perfectamente, incluyendo su fecha de entrada y su id autonumérico:

Figura 12

Este apartado es, sin lugar a dudas, el más complejo del trabajo, así como el que más tiempo ha llevado realizar. El tiempo estimado al inicio de 6 horas se ha convertido en casi 22 horas de carga de trabajo.

Ha habido varios problemas principales, entre los que destacan:

- La velocidad de transmisión en baudios de ambos canales de comunicación Serial. El módulo ESP8266 utilizado solo funciona a 115200 baudios, pero el Arduino Uno no trabaja bien con velocidades tan altas de transmisión.
- Los pines digitales del Arduino Uno, que no funcionan correctamente como Serial. Y al no disponer de un Serial dedicado, se hace complicado recibir/transmitir correctamente.
- La poca intensidad suministrada por el puerto de 3.3V de ambos Arduinos, ya que entrega 50mA y el módulo ESP8266 necesita en arranque casi 500mA, lo que en varias situaciones supone que deba reiniciarse continuamente hasta que al final se conecta correctamente. La solución habría sido utilizar una fuente de alimentación externa de 3.3V, como una una fuente específica para protoboard. Conectada al USB 3.0 del ordenador entregaría sin problema los 400-500mA necesarios.
- El comando GET para enviar al Monitor Serial requiere enviar exactamente el número de bytes especificados en el comando AT+CIPSEND. Si se envían menos o se envían más, el dato nunca llegará al archivo PHP.
- Los caracteres `\r\n` no necesitan escribirse en el comando, ya que por defecto la orden AT ya los incluye al final de cada línea.
- Cada `\r\n` es un solo caracter, no dos.

Una vez terminado este punto, se ha cumplido el objetivo:

7. Añadir al sistema una tarjeta WiFi para sustituir al Ethernet como sistema de comunicación con Internet (PEC 3 - 21 horas).

8. APLICACIÓN WEB

8.1. Montaje

El montaje en este punto no es tal, ya que no hay dispositivos físicos que añadir al sistema. Todo el 'montaje' es virtual. Se necesita mostrar en un navegador web los datos guardados en la base de datos `zzarduino`, de manera sencilla para que un gestor de personal pueda revisar las entradas que disponen del control de acceso.

Se necesita un archivo básico para esta funcionalidad, que se denominará `index.php`. En él se leerán los datos de la base de datos mediante instrucciones `SELECT` y se mostrarán en el navegador mediante código HTML.

Como archivos secundarios, se dispone del framework Jquery de Javascript para la creación de los desplegables, recarga de la página al seleccionar un filtro y el selector de fechas. También se dispondrá de un archivo CSS[14] para editar la visualización de la página web, como poner en color azul la barra superior, cambiar el tamaño de las fuentes o dibujar un cebreado gris en cada fila par de la tabla.

8.2. Código

Debido a la longitud del código y a la cantidad de archivos necesarios para esta funcionalidad, se entregan adjuntos todos ellos junto al trabajo, en una carpeta denominada '8_visor_web'.

8.3. Pruebas

El enlace a la aplicación está en la dirección:

http://modelbrush.com/miguel_arduino_trabajo/index.php

Nada más entrar en la aplicación, se muestran todos los registros que hay en la base de datos `zzarduino`:

Figura 13

[14] *Cascading Style Sheet*

El primer filtro sirve para seleccionar solo los accesos de un trabajador determinado, así como poder seleccionar los accesos erróneos seleccionando la opción `Error`:

Figura 14

El siguiente filtro permite seleccionar un rango de fechas determinado. Por defecto ambos campos se encuentran vacíos, lo que indica que se están seleccionando los registros de cualquier fecha.

Haciendo click en un cuadro de fecha se despliega un calendario:

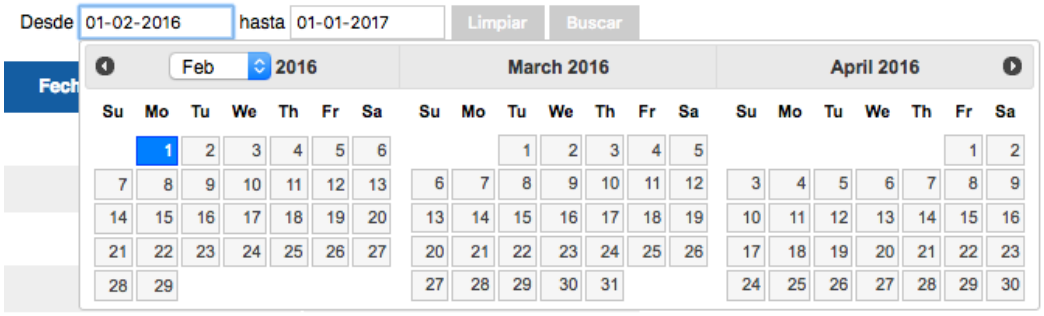


Figura 15

Una vez seleccionadas ambas fechas, o solo una de ellas, se pulsa el botón 'Buscar' para mostrar los resultados de ambos filtros, trabajador y fechas:

Figura 16

9. CONCLUSIONES

Aunque el resto del trabajo está redactado en tercera persona, estas conclusiones he decidido tratarlas en primera persona, ya que es algo más personal.

En líneas generales se han cumplido los objetivos planteados al inicio del proyecto en el punto 1.3. Objetivos del Trabajo (ver página 9). Dos objetivos he tenido que omitirlos debido al tiempo disponible.

El primero ha sido la función que enviaba un email cuando la primera persona del día entraba a trabajar y otro cuando la última persona del trabajo salía por la noche. Lo intenté, pero su complicación era elevada y debía continuar con el resto de funciones, más importantes que esta (sobretodo la función Ethernet y la de wifi).

El otro objetivo no realizado finalmente ha sido el de fabricar una caja para introducir todo el sistema y que no se viese el cableado. El resultado no estaba siendo lo esperado, y al no ser una función puramente de Arduino, decidí hacer la primera parte de las funciones secundarias opciones, el visor web. Aunque esta función tampoco es 100% de Arduino, sí que va en relación con los envíos de los datos de acceso desde el módulo Ethernet o el wifi vistos en objetivos anteriores.

Así que analizando el trabajo global, se han eliminado dos objetivos principales, aunque no demasiado importantes, y se ha realizado un objetivo opcional.

El principal defecto que he tenido a lo largo de todo el trabajo ha sido la falta de tiempo y el conocimiento casi nulo del sistema Arduino cuando comencé a realizar el trabajo. Por ello cometí el error más grave del trabajo, comprar un Arduino Leonardo en vez de un Arduino Uno. Esto supuso que llegado un punto, el Leonardo se quedase corto y tuviera que rehacer muchas cosas desde cero, tanto código como cableado. Esto me hizo perder muchos días y evitó que pudiera entregar la PEC 3 a tiempo, retrasando el resto de objetivos del trabajo.

En un punto avanzado del proyecto, en el montaje del módulo ESP8266 de wifi, el Arduino Uno se quedó también corto y tuve que adquirir un Arduino Mega, pero esto solo supuso un retraso de 2 días y apenas hubo que cambiar configuración.

Si hoy en día tuviese que comenzar el trabajo, habría adquirido un Arduino Mega en primer lugar.

Debido a todos estos problemas, la planificación se llevó a cabo correctamente hasta la PEC 3 (PEC 1 y PEC 2 se entregaron en fechas), cuya entrega retrasé 2 semanas debido a las dificultades encontradas.

El objetivo o funcionalidad más compleja del trabajo fue configurar correctamente el módulo ESP8266 de wifi, debido a las velocidades de transmisión, Seriales, comandos de consola, etc. El tiempo estipulado para esta función se disparó

hasta pasar las 20 horas, desde las 6 horas programadas (ahora sé que programadas sin ningún conocimiento de esta funcionalidad, por supuesto). El resto de objetivos sí que cumplieron básicamente los tiempo estipulados. Todos menos el objetivo de conexión a Ethernet (y envío a base de datos), que llevó algunas horas más y sobretodo el punto ya comentado del módulo wifi, que ha sido el quebradero de cabeza principal del trabajo junto con la mala elección del modelo de Arduino.

La metodología seguida en el trabajo ha sido la correcta, y la división de cada punto entre Montaje, Código y Pruebas ha demostrado ser cómoda de llevar a cabo y organizada a la hora de escribirlo. Creo que también lo es de cara al lector.

Como partes o funcionalidades que se han quedado pendientes, me habría gustado mucho poder haber probado el sistema completo durante dos semanas en mi lugar de trabajo (una tienda) para poder ver realmente la función de un control de acceso mediante NFC en una pequeña cadena de tiendas. El sistema completo, si solo hubiera adquirido uno de los Arduinos (el Mega), ha tenido un coste de aproximadamente 50€ dividido en:

Figura 17

Producto	**Cantidad**	**Precio IVA Inc.**
Arduino Mega AT2560	1	21,10€
Cable Dupont M/H	20	0,4€
Cable Dupont H/H	20	0,4€
Cable Dupont M/M	30	0,6€
Protoboard pequeña	1	1,8€
NFC RFID-RC522 (con llavero y tarjeta)	1	8,99€
Buzzer Pasivo Sunfounder	1	3,99€
Teclado numérico 4x4 membrana	1	4,95
Ethernet Shield HR911105A	1	5,56€
Pantalla LCD 16x2	1	1,61€
Módulo I2c para pantalla LCD	1	1,04€
LED alta potencia verde	1	0,1€
Módulo wifi ESP8266	1	2,1€
TOTAL	**80**	**52,64€**

Por lo tanto se ve que es muy factible instalar un dispositivo completo en cada una de las tiendas de la marca, sobretodo porque este precio bajaría aún más, ya que, por ejemplo, no sería necesario montar a la vez el Shield Ethernet y el módulo ESP8266, ahorrando otros 5€ del Shield Ethernet.

Se pueden encontrar sistemas comerciales por debajo de este precio, pero simplemente sirven para realizar la apertura de una puerta o similar, no guardan datos del acceso ni los envían inalámbricamente a una base de datos.

10. GLOSARIO

1. *Arduino* - compañía de hardware libre y una comunidad tecnológica que diseña y manufactura placas computadora de desarrollo de hardware y software, compuesta respectivamente por circuitos impresos que integran un microcontrolador y un entorno de desarrollo (IDE), en donde se programa cada placa.
2. *Baudio* - Unidad de medida utilizada en telecomunicaciones, que representa el número de símbolos por segundo en un medio de transmisión digital. Cada símbolo puede comprender 1 o más bits, dependiendo del esquema de modulación.
3. *CSS* - Hojas de estilo en cascada (o CSS, siglas en inglés de Cascading Stylesheets) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado . Es muy usado para establecer el diseño visual de las páginas web e interfaces de usuario escritas en HTML.
4. *Ethernet* - Estándar de redes de área local para ordenadores con acceso al medio por detección de la onda portadora y con detección de colisiones (CSMA/CD). Su nombre viene del concepto físico de ether. Ethernet define las características de cableado y señalización de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI.
5. *Fritzing* - Programa libre de automatización de diseño electrónico que busca ayudar a diseñadores y artistas para que puedan pasar de prototipos (usando, por ejemplo, placas de pruebas) a productos finales.
6. *HTML* - Siglas en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros.
7. *IDE* - Entorno de desarrollo integrado, en inglés Integrated Development Environment (IDE), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.
8. *IP* - Número que identifica, de manera lógica y jerárquica, a una Interfaz en red de un dispositivo que utilice el protocolo IP (Internet Protocol), que corresponde al nivel de red del modelo TCP/IP

9. *Javascript* - Lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
10. *Jquery* - Biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.
11. *LED* - Light-emitting diode (diodo emisor de luz) es un componente optoelectrónico pasivo y, más concretamente, un diodo que emite luz.
12. *LCD* - Pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora.
13. *MySQL* - Sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation, considerada como la base datos open source más popular del mundo.
14. *NFC* - Near field communication (comunicación de campo cercano) es una tecnología de comunicación inalámbrica, de corto alcance y alta frecuencia que permite el intercambio de datos entre dispositivos.
15. *PIN* - Personal Identification Number, es un número de identificación personal utilizado en ciertos sistemas, como el teléfono móvil o el cajero automático, para identificarse y obtener acceso a un sistema.
16. *PHP* - Lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.
17. *SQL* - Structured Query Language (lenguaje de consulta estructurada) es un lenguaje específico del dominio que da acceso a un sistema de gestión de bases de datos relacionales que permite especificar diversos tipos de operaciones en ellos.
18. *UIP* - Identificador único (UID) es cualquier identificador que se garantiza que sea único entre todos los identificadores utilizados para esos objetos y para un propósito específico.
19. *Wifi* - Mecanismo de conexión de dispositivos electrónicos de forma inalámbrica. Los dispositivos habilitados con wifi (como un ordenador, un televisor inteligente, una videoconsola, un teléfono inteligente o un reproductor de música) pueden conectarse a internet a través de un punto de acceso de red inalámbrica. Dicho punto de acceso tiene un alcance de unos veinte metros en interiores, distancia que es mayor al aire libre.

11. BIBLIOGRAFÍA

En esta bibliografía se muestran las páginas web que han aportado información a la hora de realizar el presente trabajo. Debido a que la información encontrada en cada página normalmente ha servido para un objetivo concreto, el número entre corchetes indica el punto del trabajo para el cual ha servido la información.

[7] Prometec, *Usando el módulo wifi ESP8266* **[web]**. Disponible en: http://www.prometec.net/esp8266/

[7] Vaupel, *ESP8266 Post data to a website/sql db* **[web]**. Disponible en: https://forum.arduino.cc/index.php?topic=367383.0

[7] Prometec, *Arduino y wifi ESP8266* **[web]**. Disponible en: http://www.prometec.net/arduino-wifi/

[6] Duds, *Arduino and MySQL with PHP - Part 3* **[web/vídeo]**. Disponible en: https://www.youtube.com/watch?v=-WlVMrEOpv8

[6] Epais, *PART 1 - Send Arduino data to the Web (PHP/ MySQL/ D3.js)* **[web]**. Disponible en: http://www.instructables.com/id/PART-1-Send-Arduino-data-to-the-Web-PHP-MySQL-D3js/

[6] Am7, *Connecting Arduino to MySQl database w/ USB using MysqlIO* **[web]**. Disponible en: http://www.instructables.com/id/Connecting-Arduino-to-MySQl-database-w-USB-using-M/step3/Setting-up-the-database-for-arduino/

[6] Jecrespom, *Librería Ethernet. Shield Ethernet y W5100* **[web]**. Disponible en: https://aprendiendoarduino.wordpress.com/2014/11/18/tema-6-comunicaciones-con-arduino-2/

[2] Aritro Mukherjee, *Security Access Using RFID Reader* **[web]**. Disponible en: https://create.arduino.cc/projecthub/Aritro/security-access-using-rfid-reader-f7c746?ref=search&ref_id=nfc&offset=4

[2] Vincent Wong, *Toggle LED with NFC Tag and PIN* **[web]**. Disponible en: https://www.hackster.io/wesee/toggle-led-with-nfc-tag-and-pin-57f894

[2] Arduhobby, *Control de Acceso mediante un modulo Lector RFID* **[web]**. Disponible en: http://arduteca.blogspot.com.es/2014/09/control-de-acceso-mediante-un-modulo_22.html

[1-7] Arduino Group, *Language Reference* **[web]**. Disponible en: https://www.arduino.cc/en/Reference/HomePage

[4] Arduino Group, *"Hello World!"* **[web]**. Disponible en: https://www.arduino.cc/en/Tutorial/HelloWorld

[3] SunFounder, *Lesson 22 Buzzer* [**web**]. Disponible en: https://www.sunfounder.com/learn/Sensor-Kit-v2-0-for-Arduino/lesson-22-buzzer-sensor-kit-v2-0-for-arduino.html

[1] Arduino Group, *Wire Library* [**web**]. Disponible en: https://www.arduino.cc/en/Reference/Wire

[5] Anita, *Arduino: Library for Multiple Keypads* [**web**]. Disponible en: http://blog.hobbycomponents.com/?p=97

[6-7] Andrew, *Logging your environment with an arduino and a data logger shield* [**web**]. Disponible en: http://blog.hobbycomponents.com/?p=215

[6] Gianmarco Randazzo, *Arduino + ethernet shield + rfid + buzzer + display + postgresql server* [**web/vídeo**]. Disponible en: https://www.youtube.com/watch?-v=iYqVH9iIy3g

[6-7] Jonathan Martín, *Sensor de temperatura con arduino y MySQL* [**web**]. Disponible en: https://www.youtube.com/watch?v=oH7WXfz_wPc

[7] RaviP6, *Esp8266 firmware update* [**web**]. Disponible en: http://www.instructables.com/id/Intro-Esp-8266-firmware-update/step4/Uploading-Firmware/

[7] Abhinaba Basu, *ESP8266 Wifi With Arduino Uno and Nano* [**web**]. Disponible en: https://blogs.msdn.microsoft.com/abhinaba/2016/01/23/esp8266-wifi-with-arduino-uno-and-nano/

[7] Khalilm, *Arduino Esp8266 Post Data to Website* [**web**]. Disponible en: http://www.instructables.com/id/Arduino-Esp8266-Post-Data-to-Website/step5/Sending-the-data/

[7] Taurusek, *Arduino Due + ESP8266 01 + CIPSEND (GET HTTP)* [**web**]. Disponible en: http://forum.arduino.cc/index.php?topic=404438.0

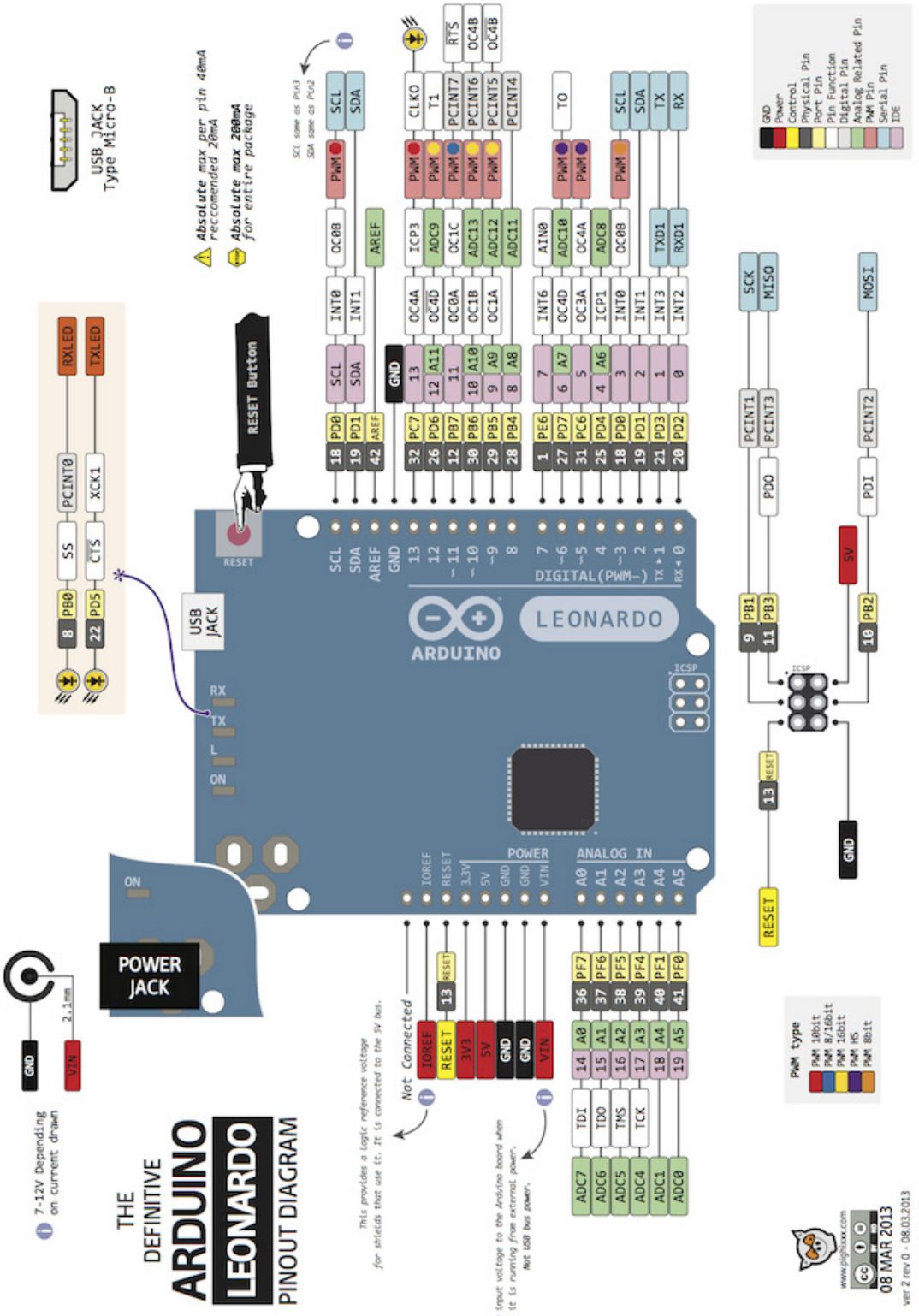
[1-8] Wikimedia Foundation, *Wikipedia. la enciclopedia libre* [**web**]. Disponible en: https://es.wikipedia.org

12. ANEXOS

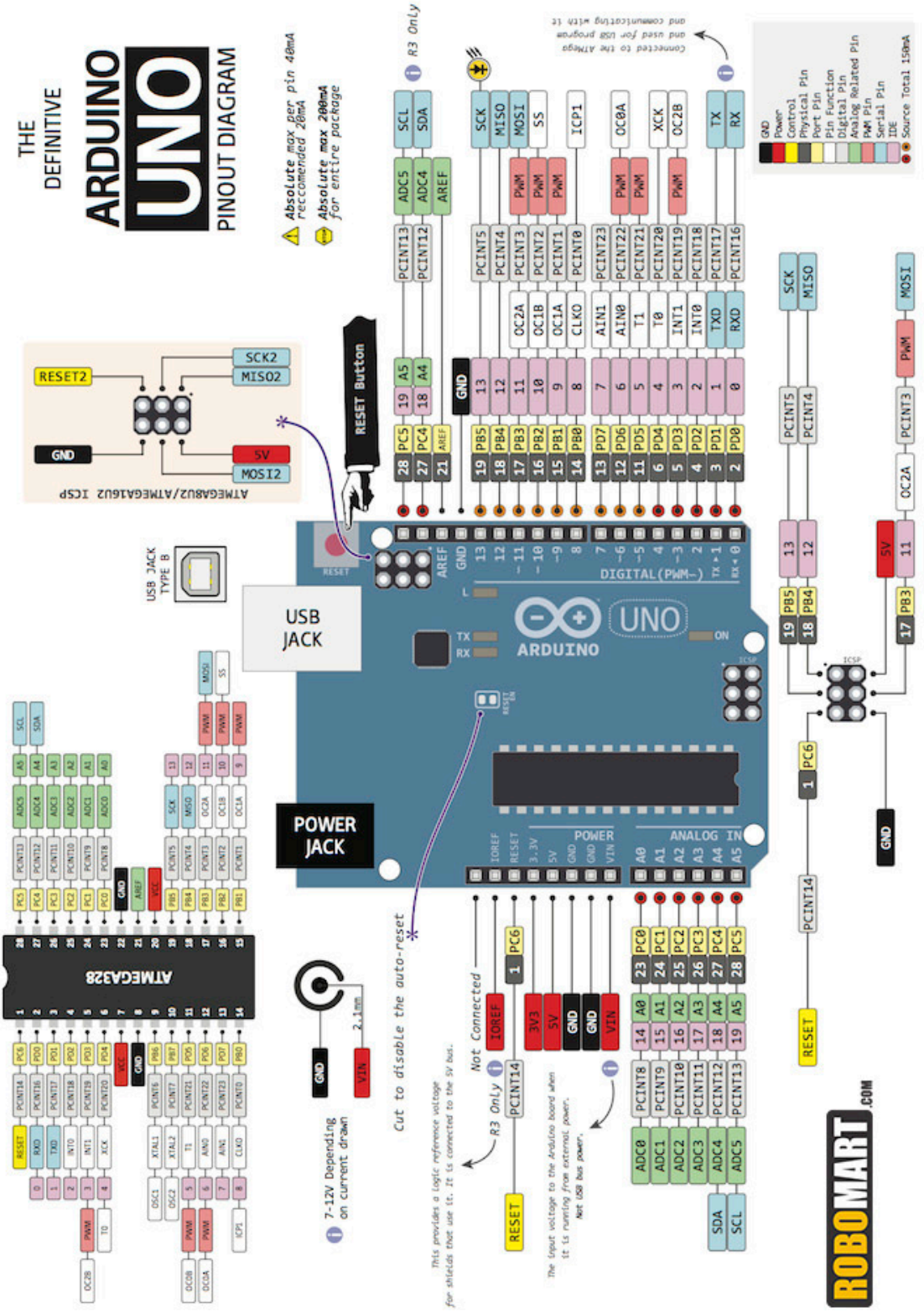
En las siguientes páginas se muestran diversos anexos:

- Anexo 1 - Esquema de puertos Arduino Leonardo
- Anexo 2 - Esquema de puertos Arduino Uno
- Anexo 3 - Esquema de puertos Arduino Mega AT2560
- Anexo 4 - Arduino Cheat Sheet

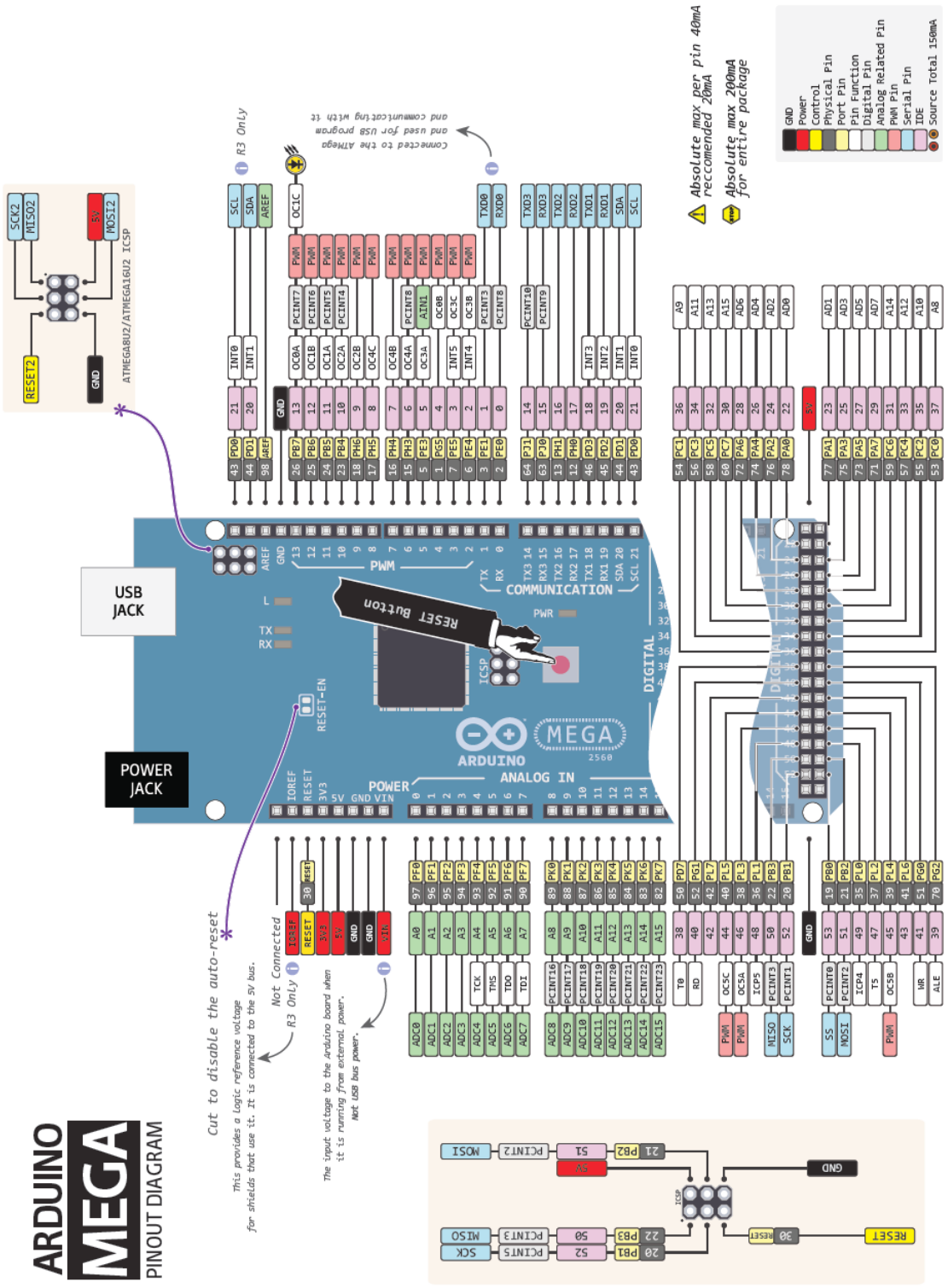
ANEXO 1 - ESQUEMA DE PUERTOS ARDUINO LEONARDO



ANEXO 2 - ESQUEMA DE PUERTOS ARDUINO UNO



ANEXO 3 - ESQUEMA DE PUERTOS ARDUINO MEGA AT2560



ARDUINO CHEAT SHEET

Content for this Cheat Sheet provided by Gavin from Robots and Dinosaurs.
For more information visit: http://arduino.cc/en/Reference/Extended

sparkfun ELECTRONICS

Structure
void **setup**() void **loop**()

Control Structures
if (x<5){ } **else** { }
switch (myvar) {
  case 1:
  break;
  case 2:
  break;
  default:
}
for (int i=0; i <= 255; i++){ }
while (x<5){ }
do { } **while** (x<5);
continue; //Go to next in do/for/while loop
return x; // Or 'return;' for voids.
goto // considered harmful :-)

Further Syntax
// (single line comment)
/* (multi-line comment) */
#**define** DOZEN 12 //Not baker's!
#**include** <avr/pgmspace.h>

General Operators
= (assignment operator)
+ (addition) - (subtraction)
* (multiplication) / (division)
% (modulo)
== (equal to) != (not equal to)
< (less than) > (greater than)
<= (less than or equal to)
>= (greater than or equal to)
&& (and) || (or) ! (not)

Pointer Access
& reference operator
* dereference operator

Bitwise Operators
& (bitwise and) | (bitwise or)
^ (bitwise xor) ~ (bitwise not)
<< (bitshift left) >> (bitshift right)

Compound Operators
++ (increment) -- (decrement)
+= (compound addition)
-= (compound subtraction)
*= (compound multiplication)
/= (compound division)
&= (compound bitwise and)
|= (compound bitwise or)

Constants
HIGH | LOW
INPUT | OUTPUT
true | false
143 // **Decimal** number
0173 // **Octal** number
0b11011111 //**Binary**
0**x**7B // **Hex** number
7**U** // Force unsigned
10**L** // Force long
15**UL** // Force long unsigned
10.0 // Forces floating point
2.4**e**5 // 240000

Data Types
void
boolean (0, 1, false, true)
char (e.g. 'a' -128 to 127)
unsigned char (0 to 255)
byte (0 to 255)
int (-32,768 to 32,767)
unsigned int (0 to 65535)
word (0 to 655word (0 to 65535)
long (-2,147,483,648 to 2,147,483,647)
unsigned long (0 to 4,294,967,295)
float (-3.4028235E+38 to 3.4028235E+38)
double (currently same as float)
sizeof(myint) // returns 2 bytes

Strings
char S1[15];
char S2[8]={'a','r','d','u','i','n','o'};
char S3[8]={'a','r','d','u','i','n','o','\0'};
//Included \0 null termination
char S4[ ] = "arduino";
char S5[8] = "arduino";
char S6[15] = "arduino";

Arrays
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};

Conversion
char() **byte**()
int() **word**()
long() **float**()

Qualifiers
static // persists between calls
volatile // use RAM (nice for ISR)
const // make read-only
PROGMEM // use flash

Digital I/O
pinMode(pin, [INPUT,OUTPUT])
digitalWrite(pin, value)
int **digitalRead**(pin)
//Write High to inputs to use pull-up res

Analog I/O
analogReference([DEFAULT, INTERNAL,EXTERNAL])
int **analogRead**(pin) //Call twice if switching pins from high Z source.
analogWrite(pin, value) // PWM

Advanced I/O
tone(pin, freqhz)
tone(pin, freqhz ,duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin, [MSBFIRST,LSBFIRST], value)
unsigned long **pulseIn**(pin,[HIGH,LOW])

Time
unsigned long **millis**() // 50 days overflow.
unsigned long **micros**() // 70 min overflow
delay(ms)
delayMicroseconds(us)

Math
min(x, y) **max**(x, y) **abs**(x)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
pow(base, exponent) **sqrt**(x)
sin(rad) **cos**(rad) **tan**(rad)

Random Numbers
randomSeed(seed) // Long or int
long **random**(max)
long **random**(min, max)

Bits and Bytes
lowByte()
highByte()
bitRead(x,bitn)
bitWrite(x,bitn,bit)
bitSet(x,bitn)
bitClear(x,bitn)
bit(bitn) //bitn: 0-LSB 7-MSB

External Interrupts
attachInterrupt(interrupt, function, [LOW,CHANGE,RISING,FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()

Libraries:

Serial.
  begin([300, 1200, 2400, 4800, 9600,14400, 19200, 28800, 38400, 57600,115200])
  end()
  int **available**()
  int **read**()
  flush()
  print()
  println()
  write()

EEPROM (#include <EEPROM.h>)
byte **read**(intAddr)
  write(intAddr,myByte)

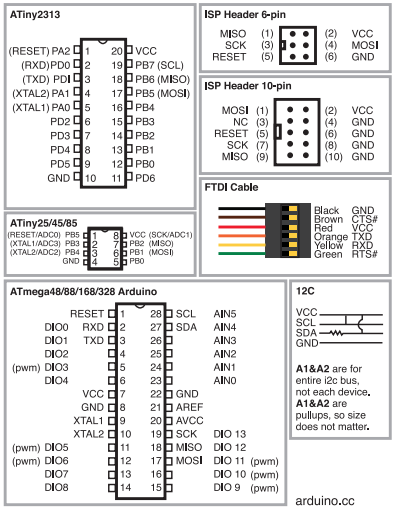
Servo (#include <Servo.h>)
  attach(pin , [min_uS, max_uS])
  write(angle) // 0-180
  writeMicroseconds(uS) //1000-2000,1500 is midpoint
  read() // 0-180
  attached() //Returns boolean
  detach()

SoftwareSerial(RxPin,TxPin)
  // #include<**SoftwareSerial**.h>
  begin(longSpeed) // up to 9600
  char **read**() // blocks till data
  print(myData) or **println**(myData)

Wire (#include <Wire.h>) // For **I2C**
  begin() // Join as master
  begin(addr) // Join as slave @ addr
  requestFrom(address, count)
  beginTransmission(addr) // Step 1
  send(mybyte) // Step 2
  send(char * mystring)
  send(byte * data, size)
  endTransmission() // Step 3
  byte **available**() // Num of bytes
  byte **receive**() //Return next byte
  onReceive(handler)
  onRequest(handler)

	ATMega168	ATMega328	ATMega1280
Flash (2k for boobtloader)	16kB	32kB	128kB
SRAM	1kB	2kB	8kB
EEPROM	512B	1kB	4kB

	Duemilanove/ Nano/ Pro/ ProMini	Mega
# of IO	14 + 6 analog (Nano has 14 + 8)	54 + 16 analog
Serial Pins	0 - RX 1 - TX	0 - RX1 1 - TX1 19 - RX2 18 - TX2 17 - RX3 16 - TX3 15 - RX4 14 - TX4
Ext Interrupts	2 - (Int 0) 1 - (Int 1)	2,3,21,20,19,18 (IRQ0 - IFQ5)
PWM Pins	5,6 - Timer 0 9,10 - Timer 1 3,11 - Timer 2	0 - 13
SPI	10 - SS 11 - MOSI 12 - MISO 13 - SCK	53 - SS 51 - MOSI 50 - MISO 52 - SCK
I2C	Analog4 - SDA Analog5 - SCK	20 - SDA 21 - SCL



arduino.cc