

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FACULTAD DE INGENIERÍA EN  
EN ELECTRICIDAD Y COMPUTACIÓN  
FIEC**

**PROYECTO SISTEMAS EMBEBIDOS**

**Integrantes del proyecto:**

Gustavo Castillo

Debbie Donoso

**Tema:**

Diseño e implementación de un sistema de control mediante monitoreo de temperatura para el acceso a una ciudadela

**II PAO 2020**

## INDICE

<b>I. ANTECEDENTES</b>	4
<b>II. OBJETIVOS</b>	7
Objetivo General	7
Objetivos Específicos	7
<b>III. DISPOSITIVOS PERIFÉRICOS</b>	7
<b>1. TERMÓMETRO INFRARROJO</b>	7
<b>1.1 Sensor Ultrasonido HC-SR04</b>	7
1.1.1 Especificaciones Técnicas	8
1.1.2 Partes del sensor ultrasónico	8
1.1.3 Funcionamiento	9
1.1.4 Precisión de medición de la distancia del sensor HC SR04	11
<b>1.2 Sensor de temperatura DS1621</b>	12
1.2.1 Pines	12
1.2.2 Rango de operación	13
<b>2. CONTROL</b>	14
<b>2.1 Tarjeta de desarrollo Arduino nano</b>	14
2.1.1 Especificaciones técnicas	14
2.1.2 Partes del Arduino Nano	15
2.1.3 Alimentación del Arduino Nano	17
2.1.4 Pines del Arduino Nano	17
<b>2.2 DS3231 Módulo reloj en tiempo real</b>	18
2.2.1 Especificaciones técnicas	18
2.2.2 Entradas del Módulo en tiempo real DS3231	19
2.2.3 Funcionamiento del Módulo en tiempo real DS3231	19
<b>2.3 Módulo Buzzer Pasivo</b>	21
2.3.1 Especificaciones técnicas	21
<b>3. PANTALLA DE CONTROL</b>	22
<b>3.1 Pantalla LCD 20x4</b>	22
3.1.1 Especificaciones técnicas	22
<b>4. PUERTA AUTOMÁTICA</b>	23
<b>4.1 Microcontrolador ATMEGA 328p</b>	23
4.1.1 Características de ATMEGA328P Microcontrolador AVR	23
4.1.2 Funcionamiento	24

4.1.3	Funcionamiento de los pines del Atmega 328P .....	24
4.2	Motor paso a paso .....	25
4.2.1	Características comunes de los motores .....	26
5.	BASE DE DATOS EN LA NUBE .....	26
5.1	Raspberry Pi .....	26
5.1.1	Especificaciones técnicas .....	27
5.1.2	Esquema de componentes y puertos .....	28
5.1.3	Raspbian .....	28
5.1.4	Base de datos relacional .....	28
5.2	CLEVER CLOUD .....	29
6.	BASE DE DATOS LOCAL .....	29
6.1	MARIA DB .....	29
6.1.1	Beneficios de MariaDb .....	30
7.	ESTADÍSTICAS .....	31
7.1	THINGSPEAK .....	31
IV.	PROTOCOLO .....	32
8.	Comunicación Serial .....	32
9.	Puerto y protocolo de comunicación I2C .....	32
10.	Velocidad del puerto I2C .....	33
V.	METODOLOGIA .....	34
11.	Esquemático en Proteus .....	34
11.1	Partes del sistema de monitoreo .....	34
12.	Pruebas de Funcionamiento .....	35
VI.	ANALISIS DE RESULTADOS .....	42
VII.	CONCLUSIONES .....	42
VIII.	RECOMENDACIONES .....	42
IX.	BIBLIOGRAFÍA .....	43
X.	ANEXOS .....	44
13.	Código Arduino .....	44
14.	Código Raspbian .....	51
15.	Código Portero para APP en Raspbian .....	53
16.	Código Atmega328p .....	53

## I. ANTECEDENTES

“No es tiempo para la comodidad y facilidad. Es tiempo para atreverse y resistir” (Churchill)

En la actualidad la distancia social juega un papel muy importante en la vida de cada persona, pero más allá de ser la distancia social un factor de suma importancia es una medida de protección, además cabe mencionar que otro factor que ha estado siempre presente en nuestras vidas es la comodidad al momento de realizar alguna acción u adquirir algún producto y es por esto que se presenta la propuesta del diseño e implementación de un sistema de control mediante monitoreo de temperatura para el acceso a una ciudadela.

A través de los años la electrónica ha ido evolucionando considerablemente, integrándose cada vez más a nuestro estilo de vida y un claro ejemplo de esto es la creación de termómetros infrarrojos pues según Sermer (2018):

“Son uno de los instrumentos que no pueden faltar en casa y menos cuando se tiene niños pequeños, debido a que la fiebre es uno de los problemas más comunes y es por esto que los termómetros son tan necesarios. En los últimos años, cada vez son más las personas que están utilizando el termómetro infrarrojo y han podido comprobar de primera mano sus beneficios”

A continuación, se presentan algunos de ellos:

1. No es necesario el contacto físico. Esto hace que sea mucho más fácil tomar la temperatura y mantener una debida distancia.
2. Son muy rápidos. Muchos de estos termómetros permiten medir la temperatura en apenas 5 segundos.
3. Fácil de usar, limpiar y desinfectar.
4. Reduce el riesgo de contaminación cruzada entre las personas evaluadas.

Girodmedical (2020) afirma que:

“Un termómetro infrarrojo está constituido por:

- Un sensor
- Un sistema óptico
- Una unidad de cálculo con algoritmos

Específicamente, la señal pasa primero por una lentilla situada en el sensor. A continuación, esta señal es amplificada y transformada proporcionalmente según la potencia hasta la temperatura del objeto medido”

Actualmente varias personas han desarrollado este tipo de termómetros mediante el uso de microcontroladores, así como el Arduino Uno o Arduino Nano, de acuerdo con Hernández (2018): “La conexión del termómetro infrarrojo con Arduino es muy sencilla ya que utiliza la interfaz de comunicación I2C como muchos otros componentes”.

Específicamente, la señal pasa primero por una lentilla situada en el sensor. A continuación, esta señal es amplificada y transformada proporcionalmente según la potencia hasta la temperatura del objeto medido”

Actualmente varias personas han desarrollado este tipo de termómetros mediante el uso de microcontroladores, así como el Arduino Uno o Arduino Nano, de acuerdo con Hernández (2018): “La conexión del termómetro infrarrojo con Arduino es muy sencilla ya que utiliza la interfaz de comunicación I2C como muchos otros componentes”.

Además del termómetro infrarrojo, otro avance tecnológico son las puertas automáticas, las cuales han sido sinónimo de comodidad para varias personas debido a que basta presionar un botón o dar la orden de abrir la puerta para poder ingresar, pero sin tener que bajar del vehículo, según Stepien & Barno (2020) las principales ventajas de estas son:

“Seguridad: es la principal ventaja de cualquier puerta automática. Cuentan con un sensor de movimiento que, en hogares y edificios, garantiza que la puerta no se cerrará si alguien está pasando y, en industrias, asegura que la mercancía no se dañará por un posible cierre inesperado de la puerta.

Comodidad: el hecho de no tener que abrir y cerrar puertas a nuestro paso aumenta la comodidad. Además, supone un ahorro de tiempo.

Accesibilidad: las puertas facilitan el paso de personas con movilidad reducida y ancianos.”

De acuerdo con Matic-port (2018):

“Las puertas automáticas tienen un funcionamiento muy simple, ya que se activan mediante un sensor de detección óptico o de movimiento, que le “informa” a la puerta de que debe abrirse o cerrarse.

Este sensor que determina cuándo la puerta se debe cerrar o abrir, y se coloca en la parte superior de la puerta, arriba y en la parte inferior o en a un lado de esta para que puedan captar el movimiento que iniciará la apertura o el cerramiento de la puerta.

El sensor óptico o de movimiento se conecta a un dispositivo de transmisión que controla un mecanismo de embrague conectado a una rueda dentada y a la puerta. Estas están conectadas por cables que realizan el movimiento de apertura y cierre de las puertas.”

Al igual que los termómetros infrarrojos también se han desarrollado puertas automáticas a través del uso de servomotores y microcontroladores, Manrique (2016) afirma que:

“Un servomotor o motor paso a paso es un motor que, ante un determinado valor de voltaje, gira a un ángulo determinado y se detiene. Podemos decir que es un motor de precisión.

Su programación en Arduino es realmente sencilla. Básicamente tenemos que vincular el servomotor a una variable de nuestro sistema y a partir de ahí ya podemos indicarle que gire a un determinado ángulo”.

Por otro lado, Gluón (2020) afirma que: “Se debe generar una señal PWM para que el servo/dispositivo funcione. Queremos que la señal tenga una frecuencia de 50Hz, es decir un periodo de 20 milisegundos, y que la parte alta de la señal dure entre 1 y 2 milisegundos. Para ello tenemos que primero, configurar el modo de operación a Fast-PWM”.

Otro avance significativo es la creación de bases de datos en la nube, este avance ha revolucionado la manera de almacenamiento y manejo de datos debido a que son una nueva modalidad de almacenamiento que difieren de las bases tradicionales. A diferencia de estas, las bases de datos en cloud no se almacenan en un equipo o sistema local, sino que se ejecuta desde la infraestructura de un proveedor de servicios.

De acuerdo con RGPD (2019) las principales ventajas de las bases de datos son:

- “Son escalables. El proveedor de servicios puede aumentar los recursos o proporcionar más espacio de almacenamiento en función de las necesidades del cliente.
- Mantienen la integridad de la información. Este tipo de almacenamiento de información puede replicar instancias de las bases de datos, permitiendo así el acceso concurrente de usuarios sin que los datos pierdan integridad.
- Ahorro de espacio físico. Las empresas que contratan este tipo de servicio no necesitan instalar infraestructuras en su empresa, todo queda almacenado en los servidores del proveedor.
- Evitan problemas. Algunos de ellos muy típicos en las bases de datos tradicionales, como la imposibilidad de acceder a la información si se caen los servidores locales.
- Aumentan la seguridad. Normalmente los proveedores de almacenamiento en la nube son empresas de reputación probada en el mundo digital e incorporan soluciones para evitar brechas de seguridad o pérdidas de información.
- Por último, se puede acceder a las bases de datos desde cualquier lugar y con cualquier dispositivo, siempre que se tenga conexión a internet y las claves de acceso.”

Las bases de datos en la nube no solo pueden ser proporcionadas por un proveedor, ahora cada vez más personas están haciendo uso de una Raspberry pi para alojar sus datos en una nube privada debido a que se pueden alojar cualquier tipo de servicios.

Teniendo en cuenta la aplicación de los distintos tipos de dispositivos tecnológicos presentados y la manera en la que se pueden desarrollar con microcontroladores ahora se hablará sobre la implementación del proyecto propuesto.

Para el presente proyecto se utilizará un microcontrolador de Arduino para el control del sensor infrarrojo mediante comunicación I2C, también se usará el microcontrolador para el control del sensor ultrasónico, pantalla LCD y buzzer, para la parte del control de la puerta se usará un servomotor y un atmega328P, finalmente para la creación de la base de datos se usará una Raspberry Pi debido a que si se tiene una nube en esta, se tiene la posibilidad de controlar la información sin ningún proveedor de por medio.

Evidentemente, si se quiere tener una nube en la Raspberry, es necesario que esta esté siempre en funcionamiento. Y la ventaja de tener una nube en la Raspberry Pi, es el bajo consumo de esta frente a otros servidores. La Raspberry Pi, tiene unas prestaciones limitadas, pero como contrapartida, tiene un reducido consumo de energía.

## II. OBJETIVOS

### Objetivo General

Diseñar un sistema de control y acceso a ciudadela mediante monitoreo de temperatura y portero automático para identificación de posibles casos covid-19

### Objetivos Específicos

- Obtener la temperatura de cada persona que ingrese a la ciudadela a través de un termómetro infrarrojo.
- Identificar personas con altas temperaturas haciendo uso de alarma y visualización de valor de temperatura
- Establecer una portería automática haciendo uso de APP en Android para que la persona encargada evite el contacto con la puerta.
- Almacenar en la nube y en una base de datos relacional la información recolectada para monitoreo mediante registros.

## III. DISPOSITIVOS PERIFÉRICOS

### 1. TERMÓMETRO INFRARROJO

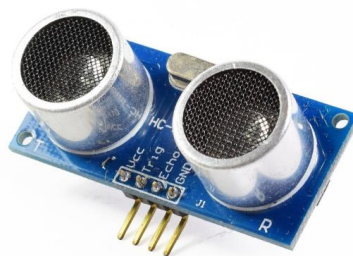
Algunos dispositivos usados para obtener la información del entorno físico son los sensores. Para la fabricación de un termómetro infrarrojo se utilizará un sensor ultrasonido, un sensor de temperatura a distancia, módulo de reloj, arduino nano, buzzer pasivo, pantalla LCD 20x4.

#### 1.1 Sensor Ultrasonido HC-SR04

El sensor ultrasonido HC-SR04 permite detectar objetos a distancia, este dispositivo determinara si hay una persona frente al termómetro, su funcionamiento está basado en ondas de ultrasonido, estas ondas de sonido son emitidas por el sensor que al interceptarse con un objeto estas regresan y el cálculo de la distancia se lo hace con el tiempo que demora la onda en regresar.

La distancia se puede calcular utilizando la siguiente formula:

$$\text{Distancia(m)} = \{(\text{Tiempo del pulso ECO}) * (\text{Velocidad del sonido}=340\text{m/s})\}/2$$



Naylampmechatronics (2020). Sensor Ultrasonido HC-SR04.Figura 1. Recuperado de <https://www.naylampmechatronics.com/>

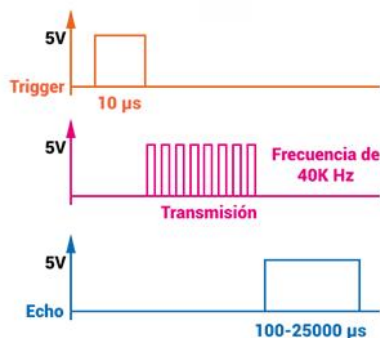
### 1.1.1 Especificaciones Técnicas

- Voltaje de Operación: 5V DC
- Corriente de reposo: < 2mA
- Corriente de trabajo: 15mA
- Rango de medición: 2cm a 450cm
- Precisión: +- 3mm
- Ángulo de apertura: 15°
- Frecuencia de ultrasonido: 40KHz
- Duración mínima del pulso de disparo TRIG (nivel TTL): 10  $\mu$ S
- Duración del pulso ECO de salida (nivel TTL): 100-25000  $\mu$ S
- Dimensiones: 45mm x 20mm x 15mm
- Tiempo mínimo de espera entre una medida y el inicio de otra 20ms (recomendable 50ms)

### 1.1.2 Partes del sensor ultrasónico

## PINOUT SENSOR ULTRASONICO HC-SR04

Rango de medición  
2 cm a 400 cm



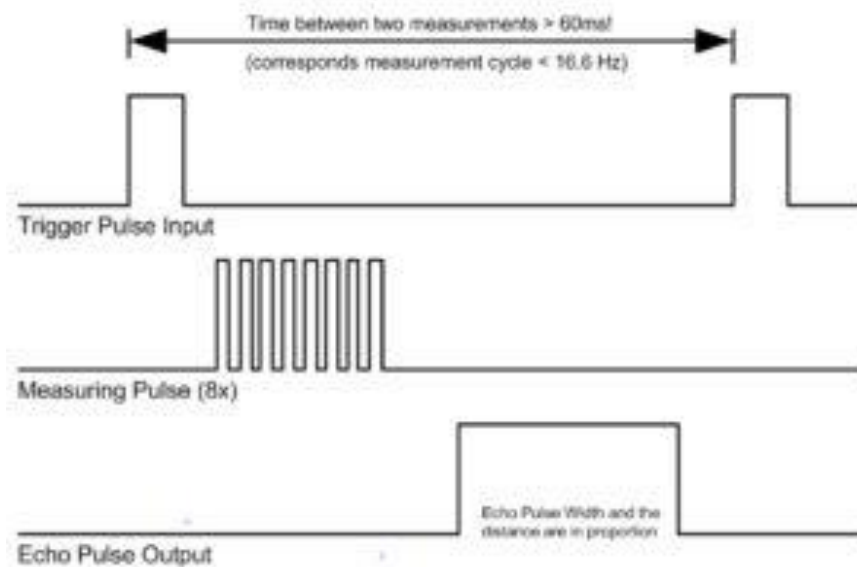
**Vcc 5 V**  
**Trigger**  
**Echo**  
**GND**



Electronics (2018). Partes del sensor ultrasonido. Recuperado de <http://uelectronics.com/wp-content/uploads/pinout-AR008.png>

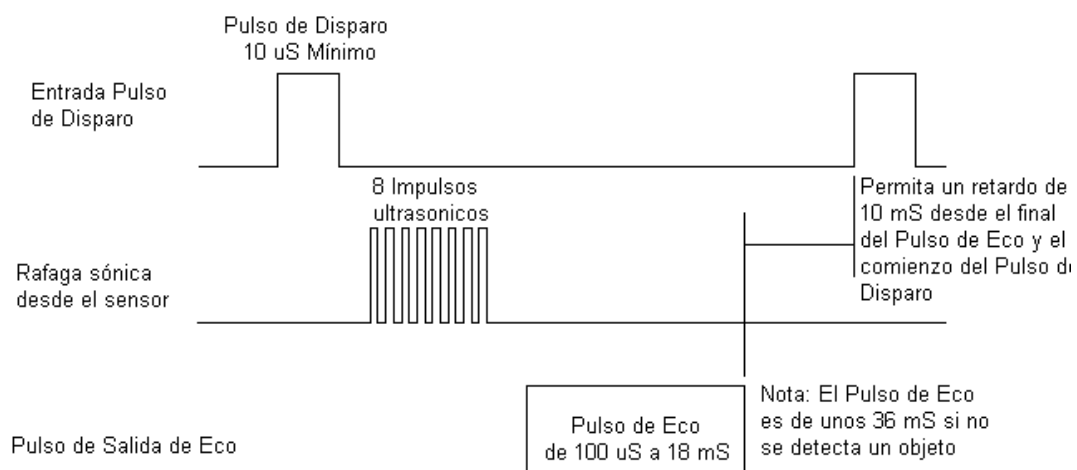
- **Contacto de potencia** +5V DC;
- **Trig** – señal de entrada;
- **Echo** – señal de salida;
- GND es la salida “Tierra”

### 1.1.3 Funcionamiento



Arduino (2018). Pulso de entrada. Recuperado de <https://proyectosconarduino.com/sensores/sensor-de-distancia-hc-sr04/>

### Diagrama de Tiempos del SRF04



INTPLUS (2020). Pulso de entrada. Recuperado de  
<http://www.superrobotica.com/S320110.htm>

Para que el sensor empiece a funcionar éste deberá recibir un pulso alto (HIGH) por el puerto del trigger el cual debe tener una duración de pulso 10  $\mu$ s (10 microsegundos) para comenzar su ciclo de detección a distancia.

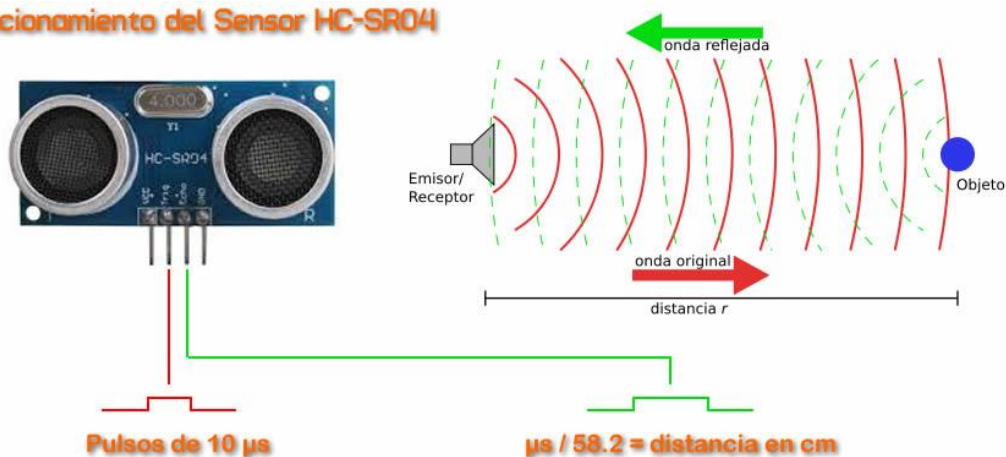
Debido a que este sensor es conectado al microcontrolador de arduino debe ser convertido en 8 pulsos con una frecuencia de 40 kHz, que serán enviados a través del trigger (Trig);

Arduino (2018) afirma que:

“Cuando los pulsos emitidos llegan a un obstáculo, rebotan y vuelven, siendo recibidos por el receptor, el cual generará un pulso alto (HIGH) en la salida (pin Echo) de igual duración que el tiempo transcurrido entre la emisión y la recepción del pulso.

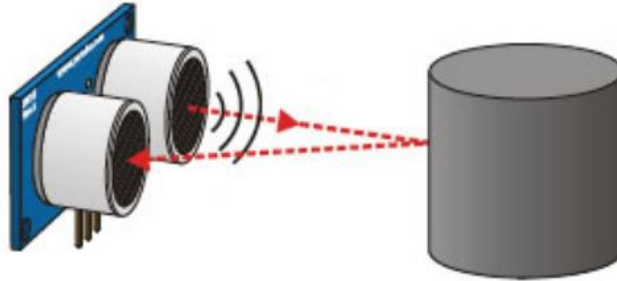
En el lado del controlador, la señal recibida debe ser convertida a distancia utilizando fórmulas. Al dividir el ancho del pulso por 58.2, obtenemos los datos en centímetros, cuando se divide por 148 – en pulgadas”.

#### Funcionamiento del Sensor HC-SR04



INTPLUS (2020). Funcionamiento. Recuperado de  
<http://www.superrobotica.com/S320110.htm>

A continuación, se puede observar como el sensor lanza un pulso y recibe una respuesta a través del “Eco” que se produce al rebotar contra una superficie sólida. En este caso, por ejemplo, una pared. Este principio es el que se utilizará para poder determinar la distancia al medir el tiempo entre la ida y la vuelta y al conocer la velocidad de propagación del sonido en el aire.



$$\text{Tiempo} = 2 * (\text{Distancia} / \text{Velocidad})$$

$$\text{Distancia} = \text{Tiempo} \cdot \text{Velocidad} / 2$$

Proserquisa (2018). Fórmula. Recuperado de <http://cursoarduino.proserquisa.com/wp-content/uploads/2016/10/Tutorial-13-Modulo-sensor-Ultrasonico.pdf>

De acuerdo con el principio descrito anteriormente se usará una fórmula sencilla para determinar la distancia, para esto basta con decir que la velocidad es igual al espacio dividido por el tiempo que se tarda en recorrer dicho espacio. La velocidad del sonido es conocida (343m/s) y el tiempo lo vamos a determinar, como el tiempo que transcurre desde que efectuamos el disparo hasta que recibimos el eco.

$$\text{Velocidad} = \frac{\text{Espacio}}{\text{Tiempo}} \longrightarrow \text{Espacio} = \text{Velocidad} \times \text{Tiempo}$$

$$\text{Velocidad del sonido} = 343 \text{ m/s} = 0.0343 \text{ cm}/\mu\text{s}$$

$$\text{Espacio} = 0.0343 \times \text{Tiempo}$$

\*Pero como la onda ha recorrido el camino dos veces (ida y vuelta) hay que dividir entre dos para conocer la distancia a la que se encuentra el objeto.

$$\boxed{\text{Espacio} = 0.01715 \times \text{Tiempo}}$$

Market (2018). Conversión. Recuperado de <https://www.zonamaker.com/arduino/modulos-sensores-y-shields/ultrasonido-hc-sr04>

#### 1.1.4 Precisión de medición de la distancia del sensor HC SR04

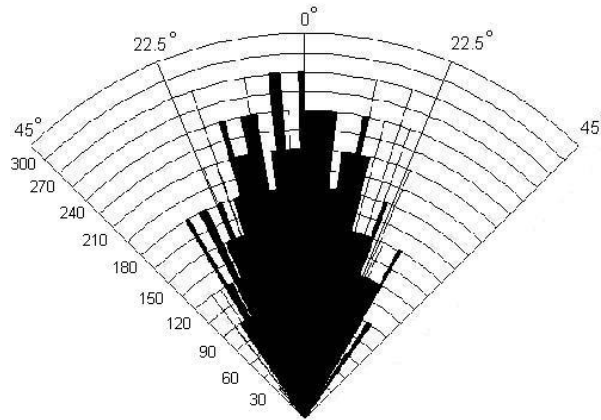
Según Arduino (2018):

“La precisión del sensor depende principalmente de dos factores:

- Temperatura y humedad del aire
- La distancia al objeto

El factor de distancia es el más importante porque la probabilidad de reflexión de los objetos vecinos aumenta y la señal misma se desvanece con la distancia.”

Cabe mencionar que el Rango efectivo del SRF04 es de unos 30° como puede verse en el siguiente diagrama:

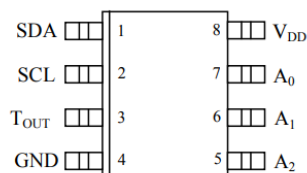


INTPLUS (2020). Rango. Recuperado de <http://www.superrobotica.com/S320110.htm>

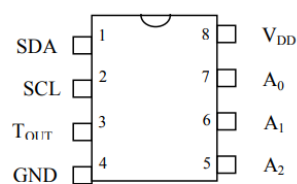
## 1.2 Sensor de temperatura DS1621

El termómetro y termostato digital DS1621 proporciona lecturas de temperatura de 9 bits, que indican la temperatura del dispositivo. La salida de alarma térmica, TOUT, está activa cuando la temperatura del dispositivo excede una temperatura definida por el usuario TH. La salida permanece activa hasta que baja la temperatura por debajo de la temperatura TL definida por el usuario, lo que permite cualquier histéresis necesaria.

### 1.2.1 Pines



DS1621S 8-PIN SO (150mil)  
DS1621V 8-PIN SO (208mil)



DS1621 8-PIN DIP (300mil)

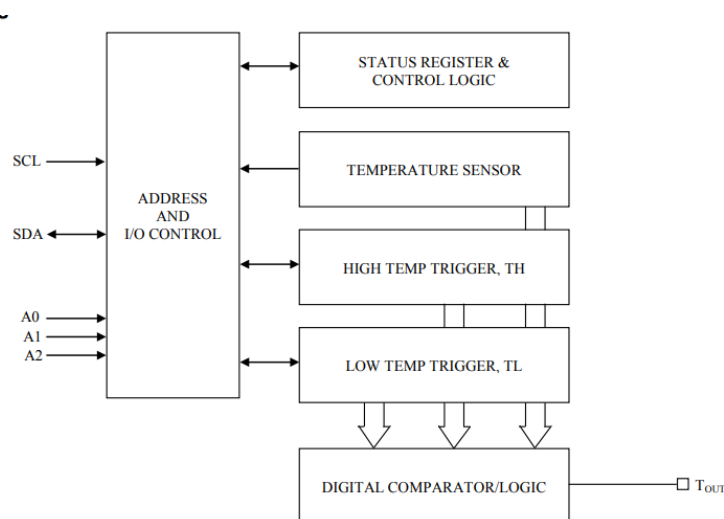
Maxim Integrated (2015). Pines. Recuperado de <https://pdfserv.maximintegrated.com/en/ds/DS1621.pdf>

- SDA: entrada / salida de datos en serie de 2 cables
- SCL - Reloj en serie de 2 hilos
- GND – Tierra
- TOUT - Señal de salida del termostato
- A0 - Entrada de dirección de chip
- A1 - Entrada de dirección de chip
- A2 - Entrada de dirección de chip
- VDD - Voltaje de la fuente de alimentación

### 1.2.2 Rango de operación

El DS1621 mide la temperatura usando un sensor de temperatura basado en bandgap. Un convertidor de analógico a digital (ADC) delta-sigma convierte la temperatura medida en un valor digital calibrado en ° C; para ° F, se debe utilizar una tabla de búsqueda o una rutina de conversión.

La lectura de temperatura se proporciona en una lectura de complemento a dos de 9 bits mediante la emisión de READ Mando TEMPERATURA. Los datos se transmiten a través de la interfaz en serie de 2 cables, MSB primero. El DS1621 puede medir la temperatura en el rango de -55 ° C a + 125 ° C en incrementos de 0.5 ° C.



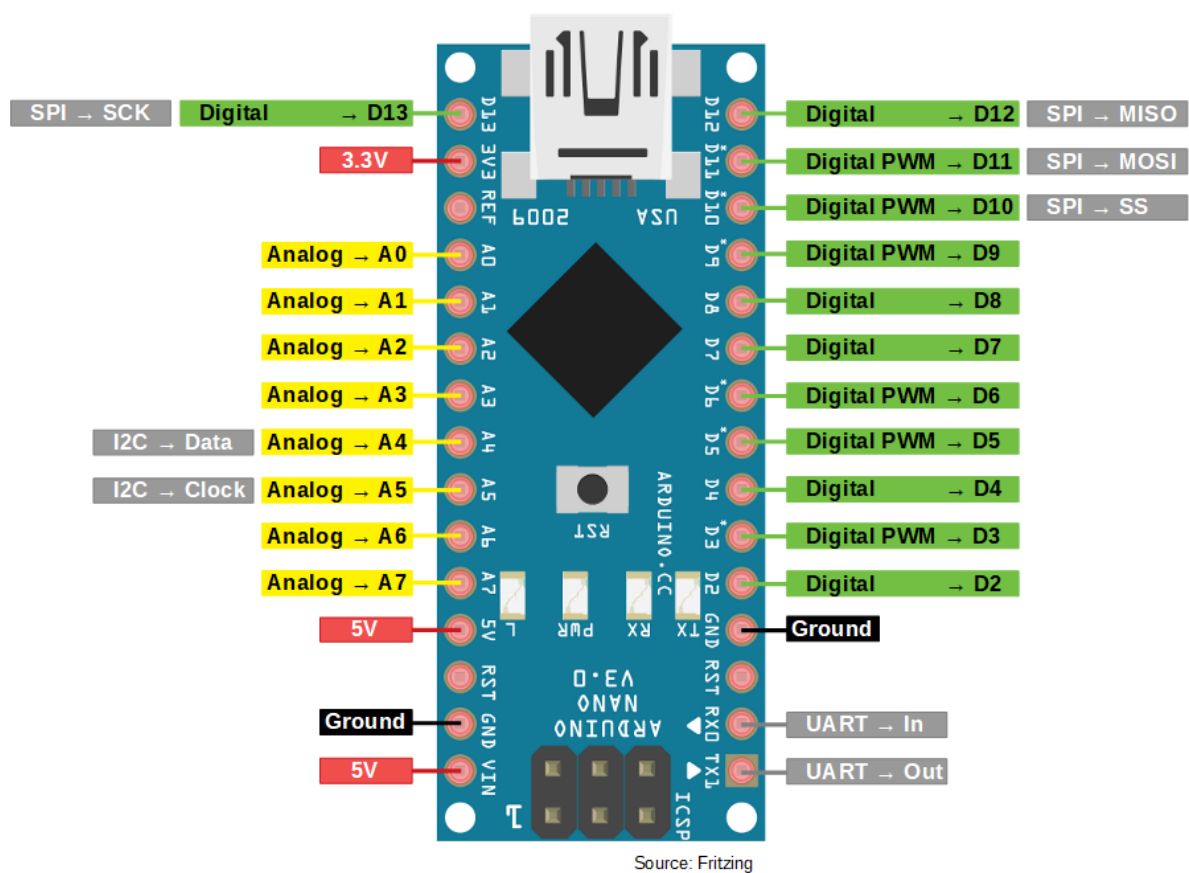
Maxim Integrated (2015). Diagrama de bloques del DS1621. Recuperado de <https://pdfserv.maximintegrated.com/en/ds/DS1621.pdf>

## 2. CONTROL

### 2.1 Tarjeta de desarrollo Arduino nano

De acuerdo con Electronilab (2018):

“El Arduino Nano es una pequeña y completa placa basada en el ATmega328 (Arduino Nano 3.0) o el ATmega168 en sus versiones anteriores (Arduino Nano 2.x) que se usa conectándola a una protoboard. Tiene más o menos la misma funcionalidad que el Arduino Duemilanove, pero con una presentación diferente. No posee conector para alimentación externa, y funciona con un cable USB Mini-B”



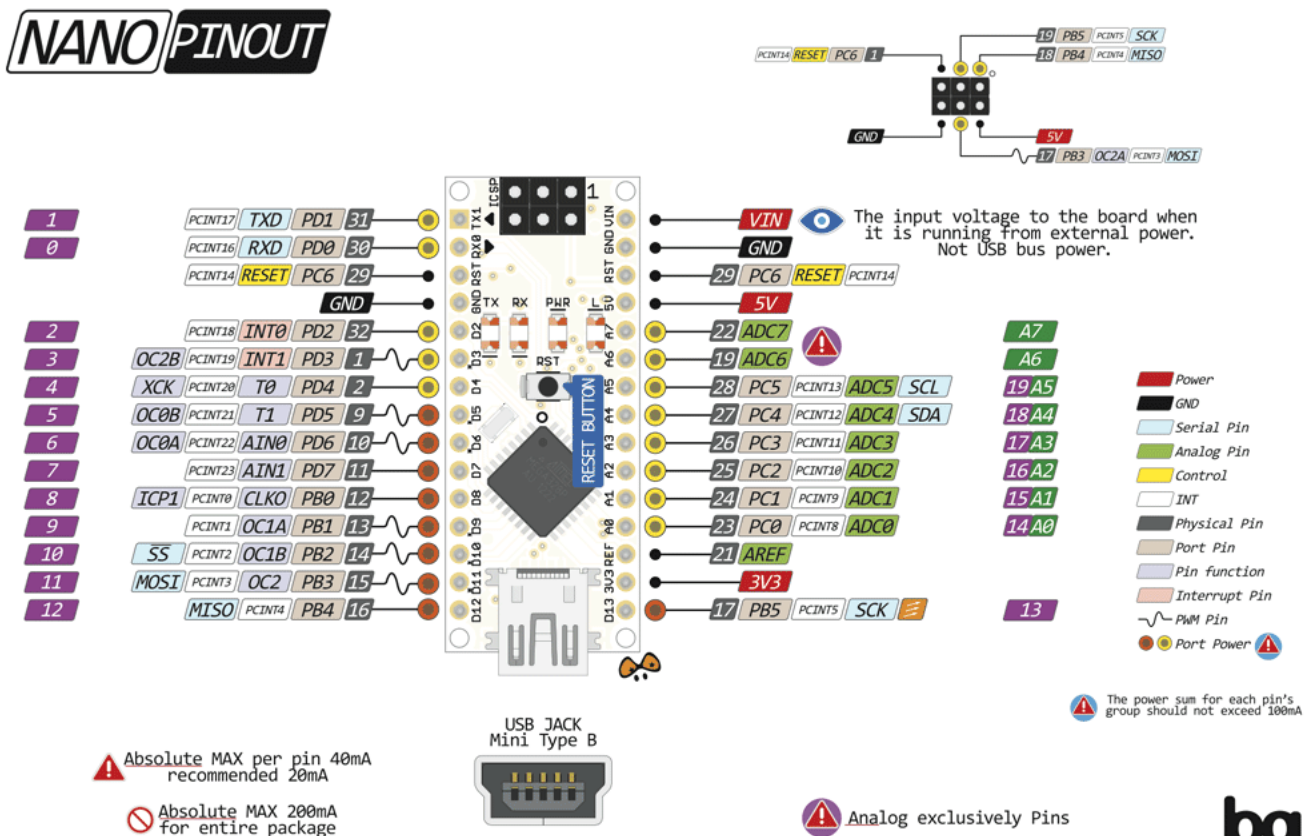
diyi0t (2020) Arduino nano. Recuperado de <https://diyi0t.com/arduino-nano-tutorial/>

#### 2.1.1 Especificaciones técnicas

- Microcontrolador Arduino ATmega328
- arquitectura, AVR

- Voltaje de operación, 5 V
- Memoria flash, 32 KB de los cuales 2 KB utilizados por bootloader
- SRAM 2 KB
- Velocidad del reloj 16 MHz
- Pines de E/S analógicas, 8
- EEPROM, 1 KB
- Corriente continua por pin entrada salida, 40 mA (Pines de E/S)
- Voltaje de entrada, 7-12 V
- Pines de E/S digitales, 22
- Salida PWM, 6
- Consumo de energía, 19 mA
- Tamaño de la placa de circuito impreso, 18 x 45 mm
- Peso, 7 g

### 2.1.2 Partes del Arduino Nano



Components (2018). Pines del Arduino Nano. Recuperado de <https://components101.com/microcontrollers/arduino-nano>

Categoría de pin	Nombre de PIN	Detalles
Poder	<b>Vin, 3,3 V, 5 V, GND</b>	<b>Vin:</b> Voltaje de entrada a Arduino cuando se usa una fuente de alimentación externa (6-12 V). <b>5V:</b> fuente de alimentación regulada que se utiliza para alimentar el microcontrolador y otros componentes de la placa. <b>3.3V:</b> suministro de 3.3V generado por el regulador de voltaje incorporado. El consumo máximo de corriente es de 50 mA. <b>GND:</b> clavijas de tierra.
Reset	<b>Reset</b>	Reinicia el microcontrolador.
Pines analógicos	<b>A0 - A7</b>	Se utiliza para medir voltaje analógico en el rango de 0-5 V
Pines de entrada / salida	<b>Pines digitales D0 - D13</b>	Se puede utilizar como pines de entrada o salida. 0V (bajo) y 5V (alto)
Serial	<b>Rx, Tx</b>	Se utiliza para recibir y transmitir datos en serie TTL.
Interrupciones externas	<b>2, 3</b>	Para activar una interrupción.
PWM	<b>3, 5, 6, 9, 11</b>	Proporciona salida PWM de 8 bits.
SPI	<b>10 (SS), 11 (MOSI), 12 (MISO) y 13 (SCK)</b>	Se utiliza para la comunicación SPI.
LED incorporado	<b>13</b>	Para encender el LED incorporado.
IIC	<b>A4 (SDA), A5 (SCA)</b>	Se utiliza para la comunicación TWI.
AREF	<b>AREF</b>	Para proporcionar voltaje de referencia para voltaje de entrada.

Components (2018). Tabla del funcionamiento de cada puerto del Arduino Nano. Recuperado de <https://components101.com/microcontrollers/arduino-nano>



### 2.1.3 Alimentación del Arduino Nano

Existen tres formas:

**Conector USB:** se debe conectar el conector mini USB a un cargador de teléfono o computadora a través de un cable y consumirá la energía necesaria para que la placa funcione

**Pin Vin:** el pin Vin se puede suministrar con 6-12 V no regulados para alimentar la placa. El regulador de voltaje incorporado lo regula a + 5V

**Pin + 5V:** Si se tiene un suministro regulado de + 5V, puede proporcionarlo directamente desde el pin + 5V del Arduino.

### 2.1.4 Pines del Arduino Nano

**De entrada y salida:**

Hay un total de 14 pines digitales y 8 pines analógicos en la placa Nano. Los pines digitales se pueden usar para interconectar sensores usándolos como pines de entrada o controlando cargas usándolos como pines de salida. Se puede utilizar una función simple como pinMode () y digitalWrite () para controlar su funcionamiento. El voltaje de operación es 0V y 5V para pines digitales. Los pines analógicos pueden medir voltaje analógico de 0 V a 5 V usando cualquiera de los 8 pines analógicos usando una función simple como analogRead ()

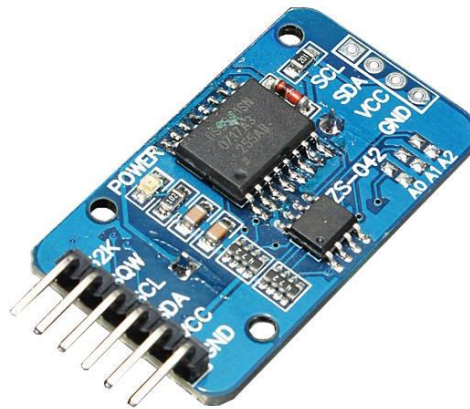
**Estos pines, además de cumplir su propósito, también se pueden usar para fines especiales que se describen a continuación:**

- **Pines seriales 0 (Rx) y 1 (Tx):** Los pines Rx y Tx se utilizan para recibir y transmitir datos seriales TTL. Están conectados con el correspondiente chip serial ATmega328P USB a TTL.
- **Pines de interrupción externa 2 y 3:** Estos pines se pueden configurar para activar una interrupción en un valor bajo, un flanco ascendente o descendente o un cambio de valor.
- **Pines 3, 5, 6, 9 y 11 de PWM:** estos pines proporcionan una salida PWM de 8 bits mediante la función analogWrite ().
- **Pines SPI 10 (SS), 11 (MOSI), 12 (MISO) y 13 (SCK):** estos pines se utilizan para la comunicación SPI.
- **LED incorporado Pin 13:** Este pin está conectado con un LED incorporado, cuando el pin 13 está ALTO - el LED está encendido y cuando el pin 13 está BAJO, está apagado.
- **I2C A4 (SDA) y A5 (SCA):** se utilizan para la comunicación IIC mediante la biblioteca Wire.
- **AREF:** Se utiliza para proporcionar voltaje de referencia para entradas analógicas con función analogReference ().
- **Restablecer pin:** al hacer que este pin sea BAJO, restablece el microcontrolador.

## 2.2 DS3231 Módulo reloj en tiempo real

El DS3231 es un reloj en tiempo real de alta exactitud que cuenta con un oscilador a cristal con compensación de temperatura (TCXO).

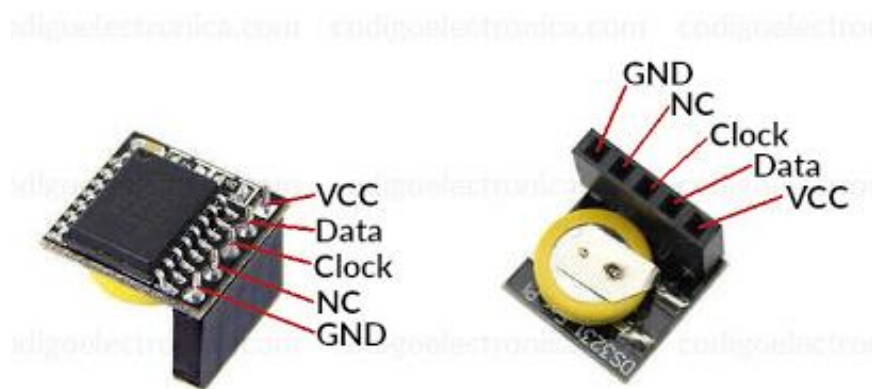
El módulo se comunica con el microcontrolador a través del bus I2C con solamente 2 pines que pueden ser compartidos por varios dispositivos como memorias EEPROM, expansores de IO, controladores PWM, etc.



Geek (2019) modulo reloj/calendario Recuperado de <https://www.geekfactory.mx/tienda/modulos-para-desarrollo/ds3231-modulo-reloj-en-tiempo-real/>

### 2.2.1 Especificaciones técnicas

- Voltaje de alimentación de 3.0 a 5 volts.
- RTC de alta exactitud, maneja todas las funciones para el mantenimiento de fecha/hora.
- Exactitud de  $\pm 2$ ppm operando a una temperatura de  $0^{\circ}\text{C}$  a  $+40^{\circ}\text{C}$ .
- Módulo cuenta con reloj DS3231 y memoria EEPROM I2C.
- El módulo cuenta con batería de respaldo (incluida).
- Registro de segundos, minutos, horas, día de la semana, fecha, mes y año con compensación de años bisiestos hasta 2100.
- El DS3231 Incluye sensor de temperatura con exactitud de  $\pm 3$  grados centígrados.
- 2 alarmas programables por hora/fecha.
- Salida de señal cuadrada programable.



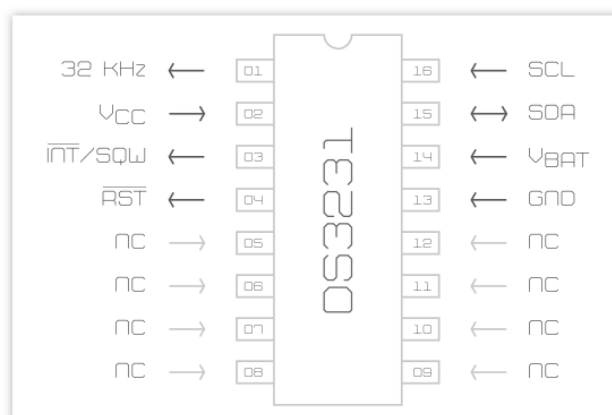
Fernández (2019). Pines del módulo en tiempo real. Recuperado de <http://codigoelectronica.com/blog/arduino-ds3231?fbclid=IwAR1faew0b8YkRg1Gfi77eGKdq9kb2h7KRF-XvGbbD3RVS1CVdeAmAZ8gLfY>

### 2.2.2 Entradas del Módulo en tiempo real DS3231

- Vcc - Contacto de potencia +5V Dc;
- GND – Salida de “Tierra”;
- Clock – señal de entrada “Pulsos del reloj que sincronizan el sistema”
- Data – Línea por la que se mueven los datos
- NC – Contacto cerrado (deja pasar la corriente mientras el dispositivo se halla en reposo)

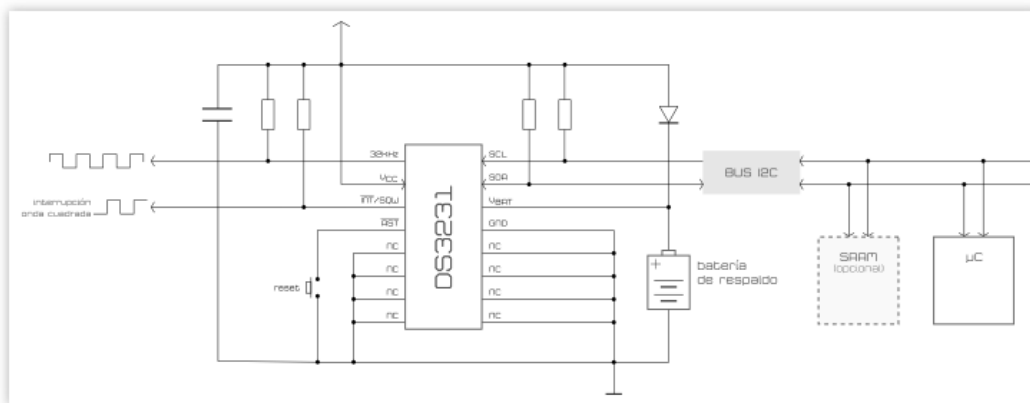
### 2.2.3 Funcionamiento del Módulo en tiempo real DS3231

El Módulo DS3231 a diferencia de otros módulos de tiempo real que usan un oscilador externo para determinar la hora, éste utiliza una tabla almacenada en la memoria interna del componente determina cómo compensar la hora en función de la temperatura y del tiempo de funcionamiento (la edad) del dispositivo. Con esta técnica, según la hoja de datos, se consigue una precisión de  $\pm 3.5$  ppm en el rango de temperatura industrial, de  $-40^{\circ}\text{C}$  a  $+85^{\circ}\text{C}$ , o de  $\pm 2$  ppm en el rango de temperatura comercial.



Ventura (2015). Pinout del DS3231. Recuperado de [http://polaridad.es/rtc-reloj-tiempo-real-ds3231-comunicaciones-i2c/?fbclid=IwAR1crM2yB\\_D0SMpT5p3bqEg4Pr65b5lb9SoPYB5Fhjwn5gpsQ7NIvJgueEg](http://polaridad.es/rtc-reloj-tiempo-real-ds3231-comunicaciones-i2c/?fbclid=IwAR1crM2yB_D0SMpT5p3bqEg4Pr65b5lb9SoPYB5Fhjwn5gpsQ7NIvJgueEg)

Normalmente, los relojes en tiempo real suelen incluir una pequeña memoria en la que almacenar cosas comunes en su uso, como un par de horas de alarma. Tampoco es raro necesitar más memoria para otras aplicaciones horarias algo más específicas, por lo que se suele añadir en las aplicaciones típicas o incluso internamente en algunas versiones de los integrados. Por ejemplo, el DS3232 es igual que el DS3231 que nos ocupa pero con 236 bytes SRAM que también se mantienen, como la fecha y la hora, con la batería de respaldo del dispositivo.



Ventura (2015). Circuito típico de uso del RTC del Módulo DS3231. Recuperado de [http://polaridad.es/rtc-reloj-tiempo-real-ds3231-comunicaciones-i2c/?fbclid=IwAR1crM2yB\\_D0SMpT5p3bqEg4Pr65b5lb9SoPYB5Fhjwn5gpsQ7NIvJgueEg](http://polaridad.es/rtc-reloj-tiempo-real-ds3231-comunicaciones-i2c/?fbclid=IwAR1crM2yB_D0SMpT5p3bqEg4Pr65b5lb9SoPYB5Fhjwn5gpsQ7NIvJgueEg)

Usualmente el módulo DS3231 funciona en conjunto con termómetros u otros medios de medida de temperatura ya que en su memoria interior permite trabajar con estos datos dando como resultado una señal de fecha y hora muy exacta.

## 2.3 Módulo Buzzer Pasivo

Los Sistemas embebidos como Arduino necesitan interactuar con el usuario ya sea por medios visuales o de sonidos. Un buzzer es el elemento más usado como indicador de sonido, puede usarse al presionar una tecla, reproducir melodías, alarmas.

El módulo incluye un transistor que cumple la función de amplificador de señal, de esta forma podemos conectar el módulo directamente a nuestro Arduino y no preocuparnos por dañar algún componente. El buzzer piezoeléctrico es de tipo pasivo por lo que puede reproducir melodías utilizando la función TONE de Arduino.



Naylamp (2019) Buzzer pasivo. Recuperado de <https://www.naylampmechatronics.com/interfaz-de-usuario/251-modulo-buzzer-pasivo.html>

### 2.3.1 Especificaciones técnicas

- Voltaje de Operación: 3.3V - 5V DC
- Tipo: Piezo eléctrico pasivo
- Incluye el transistor S8550
- Pines: VCC, GND y Señal

### 3. PANTALLA DE CONTROL

#### 3.1 Pantalla LCD 20x4

En muchos de nuestros proyectos necesitamos visualizar o monitorear parámetros, si bien una solución es los display de 7 segmentos, pero solo estamos limitados a valores numéricos e incluso si deseamos poner varios dígitos a nivel de hardware aumenta nuestro diseño electrónico por lo que necesitamos técnicas de multiplexado. Los LCD alfanuméricos son la solución más práctica, para este problema, empezando por su bajo consumo, diferentes tamaños disponibles, y trabaja con caracteres alfanuméricos, por lo que podemos representar cualquier texto o variable.



Dielect (2018) PANTALLA LCD 20X4 2004. Recuperado de <https://ssdielect.com/es/lcd-alfanumerico/648-qc2004a.html>

##### 3.1.1 Especificaciones técnicas

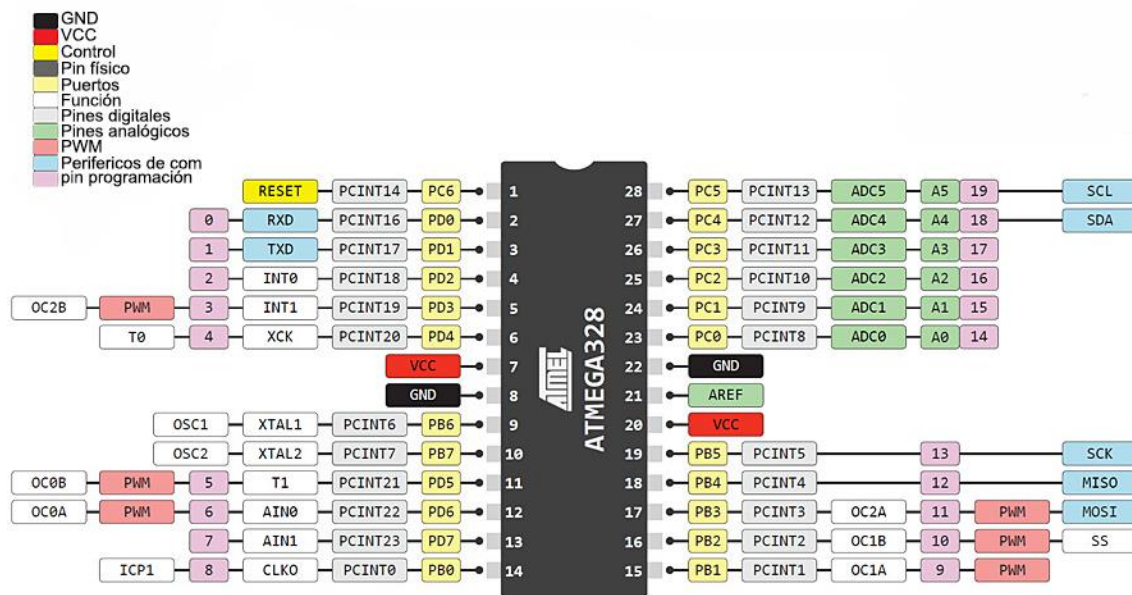
- 20 caracteres x 4 líneas
- Caracteres de 5x8 puntos
- Tamaño de caracter: 4.74 mm x 2.94 mm
- Puede mostrar letras, números, caracteres especiales, y hasta 8 caracteres creados por el usuario
- Backlight de LED color azul
- Caracteres color blanco
- Interface paralela. Puede operar en modo de 8 bits, o de 4 bits para ahorrar pines del microcontrolador
- Posee controlador KS0066 o equivalente on-board (compatible Hitachi HD44780)
- Voltaje de alimentación: 5 V
- Tamaño total: 98 mm x 60 mm x 14 mm

## 4. PUERTA AUTOMÁTICA

Para la fabricación de una puerta automática se utilizará un Atmega 328P y un servomotor.

### 4.1 Microcontrolador ATMEGA 328p

El atmega 328p es un microcontrolador con arquitectura RISC de 8 bits que sirve para controlar elementos de entrada/salida, pertenece a la familia AVR y tiene un total de 23 entradas/salidas programables, cuenta con distintos módulos como ADC, USART, PWM.



#### 4.1.1 Características de ATMEGA328P Microcontrolador AVR

- Fabricante: Atmel (Microchip).
- Voltaje de operación: 1.8 a 5.5 VDC.
- Arquitectura de CPU: 8 bit AVR
- Memoria flash: 32 KB.
- Memoria RAM: 2 KB.
- EEPROM: 2 KB.
- Frecuencia de operación: 20 Mhz.
- Pines de IO: 23
- Canales ADC: 10.
- Interfaces: UART, TWI, SPI.
- Temperatura de Operación: -40° a 85° C

#### 4.1.2 Funcionamiento

El atmega328P es un CI (circuito integrado) de alto rendimiento que está basado en un microcontrolador RISC (computador conjunto de instrucciones reducidas), con una memoria de 32KB flash con capacidad de leer entradas mientras esté ejecutando alguna acción en la parte de las salidas. Este dispositivo es uno de los microcontroladores básicos más completos que existe en la actualidad, ya que este posee una memoria EEPROM de 1KB, una memoria SRAM de 2KB, varios periféricos de comunicación (USART), (ICSP), (I2C), Entradas digitales, Entradas analógicas, Salidas digitales, Salidas PWM.

#### 4.1.3 Funcionamiento de los pines del Atmega 328P

- **VCC & GND**

Estos pines son propiamente de alimentación, donde **GND** es (0V), y **VCC** puede ser 3.3 – 5V, esta alimentación depende mucho del tipo de empaquetado que se usa (DIP), (QFP), (QFN).

- **CONTROL (RESET)**

Este pin cumple la función de reiniciar todo el proceso desde cero. Para este efecto se necesita obligatoriamente realizar las siguientes conexiones.

- **PUERTO (PB, PC, PD)**

Casi todos los pines de los  $\mu$ C están divididos en grupos o en puertos (PuertoB, puertoC, puertoD). En algunos softwares de programación es necesario saber al pie de la letra estos puertos ya que éste es la dirección física de los pines externos, en el caso del IDE de arduino no es necesario ya que estos puertos están denominados con un número específico.

- **INT0, INT1**

Estas funciones son interruptores, cuya función es interrumpir una acción o un proceso que se está ejecutando en el momento, es decir, si se tiene un cambio de estado en una de estas entradas (previamente designadas en la parte de la programación), el proceso que se está ejecutando dejará de hacer lo que está haciendo, y dará prioridad a la acción relacionada con los interruptores.

- **PINES DE ENTRADAS ANALÓGICAS (A0 -A5)**

Como su nombre lo indica estos puertos son propiamente para las entradas analógicas que tienen una resolución de 10 Bits C/U, lo que en números decimales sería 0 – 1023, con estos valores se trabajará en la parte de la programación.



- **PINES DE SALIDAS PWM**

En este dispositivo contaremos con 6 salidas PWM (modulación por ancho de pulso), que tienen una resolución de 8 Bits, lo que en números decimales serían 0 a 255, con lo cual se trabajará en la parte de la programación. A continuación, se muestra un ejemplo de control de un motor DC, donde 0 es la velocidad mínima y 255 es la velocidad máxima.

## **4.2 Motor paso a paso**

El motor paso a paso es utilizado para abrir o cerrar la puerta que permite el acceso a la ciudadela.

Los motores paso a paso (P-P) pueden verse como motores eléctricos sin sistema de conmutación. Típicamente, todas las bobinas del motor están en el estator y el rotor es, o un imán permanente o, en el caso de motores de reluctancia variables, un bloque de algún material magnéticamente blando. Toda la conmutación debe ser manejada externamente por el controlador del motor y, habitualmente, los motores y controladores están diseñados para que el motor pueda ser mantenido en una posición o rotar en uno u otro sentido. La mayoría de estos motores pueden ser manejados a frecuencias de audio permitiendo un giro rápido y, con un controlador apropiado, pueden ser arrancados y parados en posiciones controladas.

Para algunas aplicaciones existe una posibilidad de elección entre el uso de servomotores y de motores P-P. Ambos tipos ofrecen prestaciones similares para posicionamientos precisos, pero difieren en algunos aspectos. Los servomotores requieren sistemas de realimentación analógica. Típicamente, esto involucra un potenciómetro para proporcionar realimentación acerca de la posición del rotor, y alguna circuitería para dirigir corriente a través del motor de forma inversamente proporcional a la diferencia entre la posición actual y la deseada. La elección entre uno u otros tipos de motor dependen fundamentalmente de la aplicación. Por ejemplo, la repetibilidad del posicionado con un motor P-P depende de la geometría del rotor, mientras que en el servomotor generalmente depende de la estabilidad del potenciómetro y de otros componentes del circuito de realimentación.

Los motores P-P pueden ser usados en sistemas simples de control en lazo abierto. Estos son adecuados generalmente en sistemas que operan a bajas aceleraciones con cargas estáticas; el lazo cerrado puede ser esencial para aceleraciones elevadas, particularmente si involucran cargas variables. Si se sobrecarga un motor P-P en un sistema de control de lazo abierto todo el conocimiento acerca de la posición del rotor se pierde y el sistema debe ser reiniciado. Los servomotores no presentan este problema.

Los motores P-P son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos. La característica principal de estos motores es el hecho de poder moverlos un paso por cada pulso que se le aplique. Este paso puede variar desde 90° hasta pequeños movimientos de tan solo 1.8° (e incluso hasta de 0.72°), es decir, que se necesitarán 4 pasos en el primer caso (90°) y 200 para el segundo caso (1.8°), para completar un giro completo de 360° (resolución de un motor P-P).

Estos motores poseen la habilidad de poder quedar enclavados en una posición, o bien totalmente libres. Si una o más de sus bobinas está alimentada, el motor estará enclavado en la posición correspondiente y, por el contrario, quedará completamente libre si no circula corriente por ninguna de sus bobinas.

#### 4.2.1 Características comunes de los motores

Estos motores no se caracterizan únicamente por su voltaje. Las siguientes magnitudes caracterizan a un determinado motor P-P:

- **Voltaje:** éste se halla directamente impreso sobre la unidad o se especifica en su hoja de características. A veces es preciso exceder el voltaje nominal para obtener el par deseado, pero ello contribuye a un mayor calentamiento e incluso al acortamiento de la vida del motor.
- **Resistencia:** La resistencia por bobina determina la corriente del estator y, por tanto, afecta a la curva característica del par y a la velocidad máxima.
- **Resolución:** como se ha comentado anteriormente el ángulo girado en cada paso es el factor más importante de un motor P-P a efectos de una aplicación dada. La operación de medio paso dobla el número de pasos por revolución. Números grados/pasos habituales son: 0.72, 1.8, 3.6, 7.5, 15 e, incluso, 90.

## 5. BASE DE DATOS EN LA NUBE

Para la implementación de una base de datos en la nube se utilizará una Raspberry Pi.

### 5.1 Raspberry Pi

Raspberry Pi es uno de los ordenadores más básicos que podemos encontrar, también uno de los más vendidos de toda la historia informática.

Es un ordenador de placa reducida o (placa única) (SBC) de bajo costo, desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

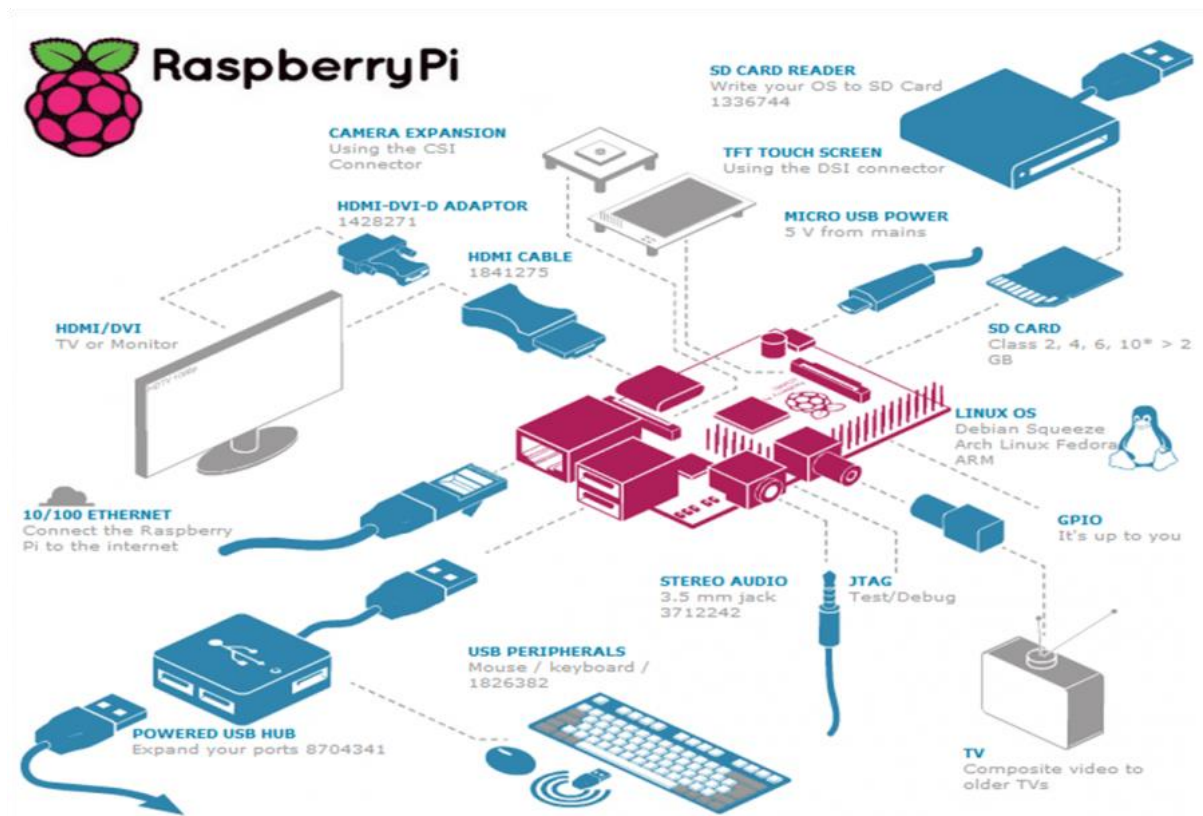


Rus (2019) Raspberry Pi. Recuperado de <https://www.xataka.com/ordenadores/raspberry-pi-4-caracteristicas-precio-ficha-tecnica>

#### 5.1.1 Especificaciones técnicas

<b>PROCESADOR</b>	ARM Cortex-A72
<b>FRECUENCIA DE RELOJ</b>	1,5 GHz
<b>GPU</b>	VideoCore VI (con soporte para OpenGL ES 3.x)
<b>MEMORIA</b>	1 GB / 2 GB / 4 GB LPDDR4 SDRAM
<b>CONECTIVIDAD</b>	Bluetooth 5.0, Wi-Fi 802.11ac, Gigabit Ethernet
<b>PUERTOS</b>	GPIO 40 pines 2 x micro HDMI 2 x USB 2.0 2 x USB 3.0 CSI (cámara Raspberry Pi) DSI (pantalla tácil) Micro SD Conector de audio jack USB-C (alimentación)

### 5.1.2 Esquema de componentes y puertos



Wiki (2019) Raspberry Pi. Recuperado de [https://smr.iesharia.org/wiki/doku.php/misc:raspbpi:caracteristicas#caracteristicas\\_generales\\_de\\_la\\_raspberry\\_pi](https://smr.iesharia.org/wiki/doku.php/misc:raspbpi:caracteristicas#caracteristicas_generales_de_la_raspberry_pi)

### 5.1.3 Raspbian

Raspbian es un sistema operativo libre recomendado para raspberry pi, se puede instalar en la cualquier versión de raspberry pi como su sistema operativo usando una tarjeta MicroSD, luego se puede proceder a la configuración en la raspberry pi.

En la terminal de raspbian se puede actualizar para tener los paquetes necesarios para la programación en lenguaje c, de esta forma se puede compilar programas y generar los ejecutables necesarios para el funcionamiento de un dispositivo.

### 5.1.4 Base de datos relacional

El modelo relacional proporciona una forma estándar de representar y consultar los datos por con el uso de tablas y técnicas de normalización que hacen que la base de datos se maneje de forma más intuitiva y eficiente.

## 5.2 CLEVER CLOUD

Clever Cloud es una empresa PaaS con sede en Europa. Ayudamos a los desarrolladores a implementar y ejecutar sus aplicaciones con una infraestructura a prueba de balas, escalamiento automático, precios justos y otras características interesantes. Nuestro objetivo es hacer un servicio fácil de usar, sin la dependencia de ningún proveedor y capaz de crecer con sus necesidades.

Clever Cloud es una plataforma de automatización de TI. Gestionamos todo el trabajo de operaciones mientras usted se centra en el valor de su negocio.



Quentin (2016).Clever Cloud. Recuperado de <https://www.slideshare.net/quentinadam/what-is-clever-cloud>

## 6. BASE DE DATOS LOCAL

### 6.1 MARIA DB

MariaDB es un sistema gestor de bases de datos (SGBD), es decir, un conjunto de programas que permiten modificar, almacenar, y extraer información de una base de datos. Disponiendo de otro tipo de funcionalidades como la administracion de usuarios, y recuperación de la información si el sistema se corrompe, entre otras.

MariaDB surge a raíz de la compra, de la compañía desarrolladora de otro (SGBD) llamado MySQL, por la empresa Sun Microsystems. El desarrollador original, decide tomar el código fuente original de MySQL y genera un derivado con mejoras y cambios a los que llama

MariaDB. Permiendo así la existencia de una versión de este producto con licencia GPL (General Public License).

### **6.1.1 Beneficios de MariaDb**

MariaDB ha seguido el desarrollo del sistema gestor MySQL, implementando diversas mejoras y nuevas funcionalidades. Las mejoras muchas veces afectan directamente al rendimiento o permiten optimizar mejor las bases de datos, por lo que usar MariaDB siempre será una opción interesante. Además, nos garantizamos que vamos a disfrutar de un software con mayor crecimiento y progresión que el propio MySQL.

Entre las novedades que ya han sido implementadas en MariaDB, podemos destacar:

- Nuevos motores de almacenamiento como Aria, que permite sustituir a MyISAM con algunas mejoras, y XtraDB, que viene a evolucionar InnoDB.
- Nuevas características disponibles, relacionadas directamente con las características disponibles en bases de datos NoSQL.
- Nueva gestión de conexiones con la base de datos, que permite multiplicar el número de accesos de manera concurrente.
- Nuevos motores de funcionamiento en cluster, como Galera, que nos permiten interesantes posibilidades de cara a la adopción Cloud.

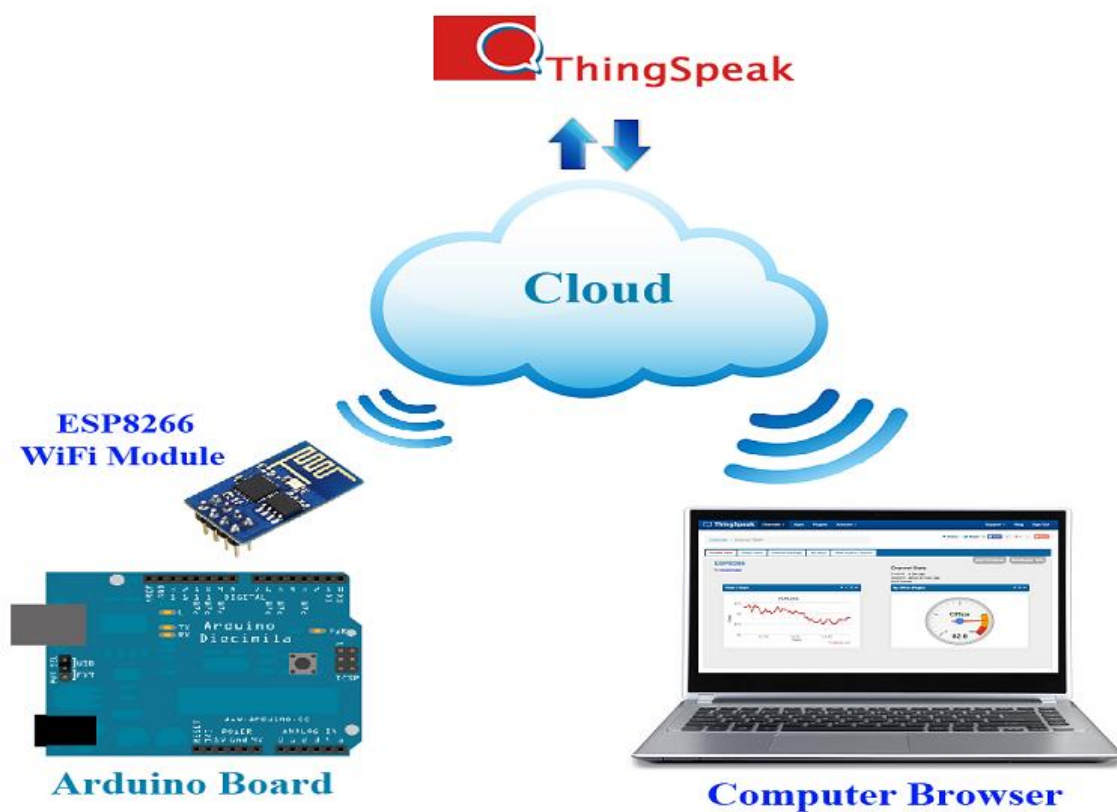
Además, al tratarse de una comunidad muy dinámica y abierta a los desarrolladores, MariaDB garantiza la aparición más rápida de parches, que puedan solucionar eventuales problemas de seguridad.

MariaDB se puede usar desde la mayoría de los sistemas de administración existentes para MySQL, como PhpMyAdmin o HeidiSQL, y es compatible con aplicaciones tan populares como WordPress, Drupal, etc. La compatibilidad es tal que muchas veces el uso de MariaDB en lugar de MySQL es transparente para desarrolladores o administradores de sistemas. Prueba de ello es que, para arrancar los servicios de MariaDB, o para hacer login en el sistema gestor por medio de línea de comandos, se usan los mismos mecanismos ya conocidos en MySQL.

## 7. ESTADÍSTICAS

### 7.1 THINGSPEAK

ThingSpeak es una plataforma de Internet of Things (IoT) que permite recoger y almacenar datos de sensores en la nube y desarrollar aplicaciones IoT. ThingSpeak también ofrece aplicaciones que permiten analizar y visualizar tus datos en MATLAB y actuar sobre los datos. Los datos de los sensores pueden ser enviados desde Arduino, Raspberry Pi, BeagleBone Black y otro HW.



ThingSpeak. Recuperado de  
<https://aprendiendoarduino.wordpress.com/2018/11/23/thingspeak/>

## IV. PROTOCOLO

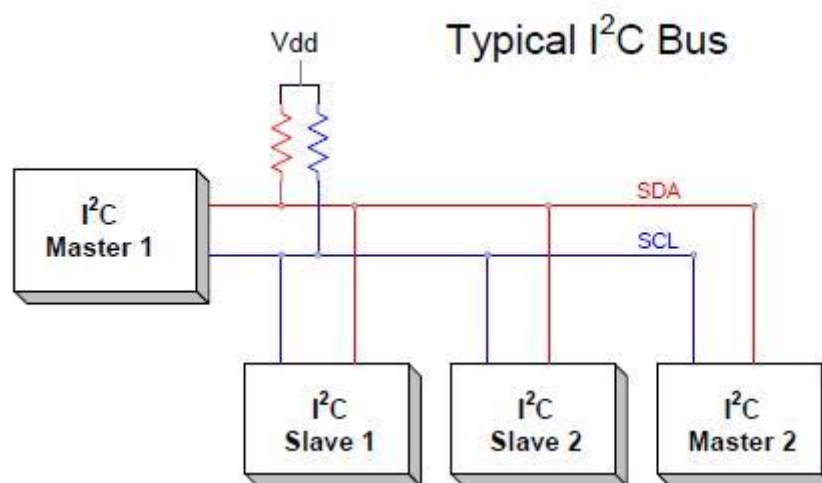
### 8. Comunicación Serial

Para la comunicación entre los distintos dispositivos se puede usar protocolos de comunicación síncrona como I2C, de esta forma se puede tener una trama de datos y definir el tipo de conexión física para el enviar y recibir datos entre dos dispositivos.

### 9. Puerto y protocolo de comunicación I2C

I2C puede ser usado en varios dispositivos que manejen información como por ejemplo sensores, codificadores, sintetizadores entre otros, al estar asociado al manejo de información, I2C permite un control sobre la información que se distribuye entre estos dispositivos que para el caso del termómetro infrarrojo los dispositivos sobre los que se usa I2C son el sensor infrarrojo, el reloj, y la pantalla LCD.

Este protocolo y puerto permite configuración de dos cables a estos dos cables están asociados los términos SDA que asocia a datos seriales y SCL que asocia a serial clock, estos dos cables forman la estructura física principal por donde se va a estar enviando y recibiendo información, los dispositivos que necesiten agregarse a la comunicación solo se anexan en una topología de bus, de esta manera se puede anexas hasta 127 dispositivos esclavos en estas dos líneas.



Configuración de bus I<sup>2</sup>C recuperado de <https://electronics.stackexchange.com/>

El bus de comunicación de la figura muestra como puede darse una configuración típica de bus entre maestro y esclavos, donde maestro proviene de un dispositivo de control como un microcontrolador y esclavo a un dispositivo como un sensor, la conexión física entre maestro y esclavo se hace a través de SDA y SCL.



# Protocolo I2C:



En la condición de inicio, el maestro envía un bit de inicio lo hace a través de la línea SDA.

En la condición de paro, la vía de SDA ahora cambia su nivel de voltaje de bajo a alto, luego de que la vía SCL cambia de bajo a alto.

Sección de dirección: es una secuencia o dirección única que van desde los 7 bits a los 10 bits, el maestro envía esta dirección a cada esclavo e identifica con que esclavo se quiere comunicar.

El maestro envía un bit de lectura/escritura, este bit es usado para indicar si el maestro necesita solicitar información del esclavo es decir necesita leer o el bit es usado para enviar información al esclavo es decir escritura.

El receptor envía el bit ACK/NACK que es usado para la verificación de envío correcto de la información, ya que luego de enviar la trama de bits el receptor valida si se entregó correctamente la información con ACK o en caso contrario con NACK.

De forma resumida se puede decir que el protocolo está estructurado de la siguiente manera:

- Iniciar la comunicación – S
- Enviar 7 bits de dirección – ADDR
- Generar 1 bit de Lectura ó Escritura – R/W
- Enviar 8 bits de dirección de memoria
- Transmitir 8 bits de datos –
- Confirmar la recepción de datos – ACK – ACKnowledged
- Generar confirmación de No-recepción, NACK – No-ACKnowledged
- Finalizar la comunicación

## 10. Velocidad del puerto I2C

La velocidad el puerto I2C se refiere al tiempo que le toma en transmitir un bit de información de forma la tasa de transferencia de información está dada en bits/s, existen tres velocidades estándar, 100Khz, 400Khz y 1Mhz, es decir, 100kbits/s, 400kbits/s y 1000kbits/s (Administrador, 2018)

## V. METODOLOGIA

### 11. Esquemático en Proteus

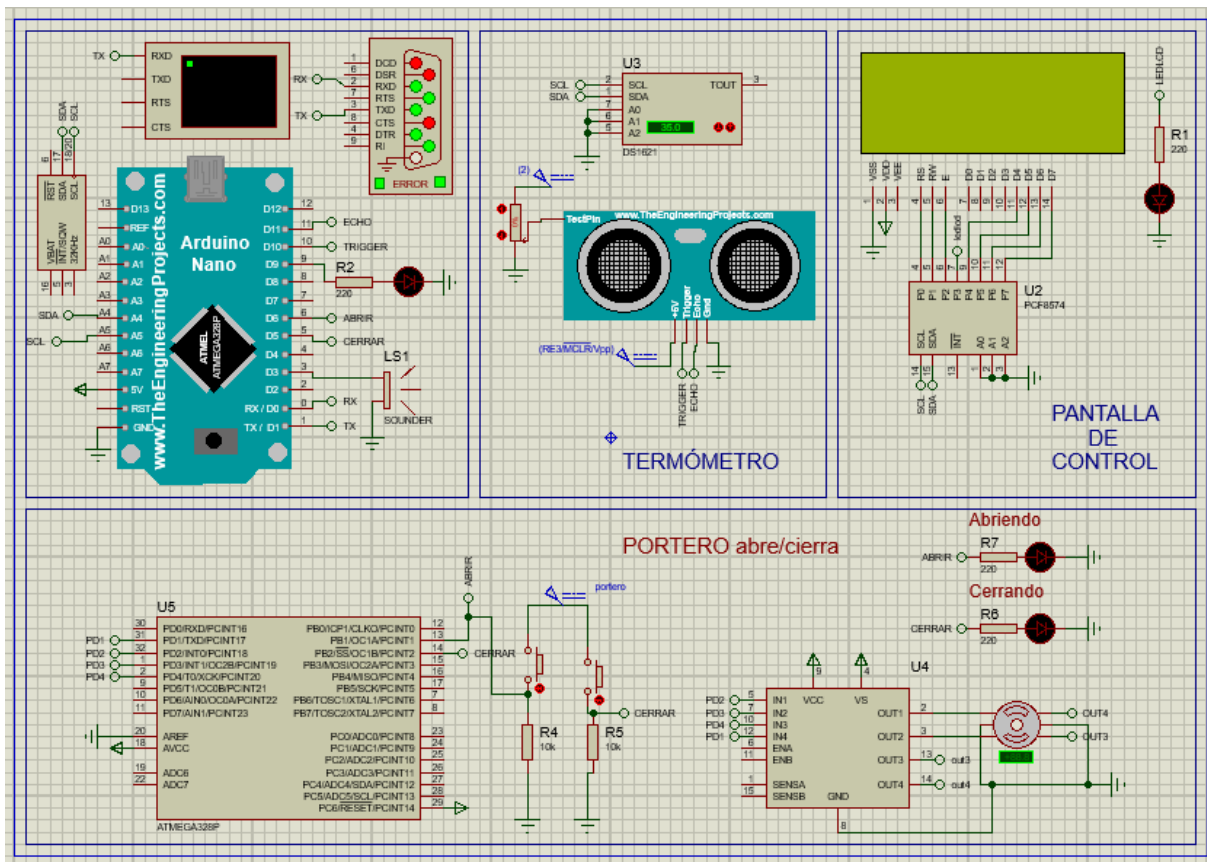


Ilustración 1 Partes del Sistema de monitoreo

#### 11.1 Partes del sistema de monitoreo

- **Control:**

En la ilustración 1 se observa la tarjeta de desarrollo Arduino nano que cumple la función de maestro en la topología Maestro-Esclavo para el protocolo I2C por lo que el sensor de temperatura, la pantalla LCD y el reloj son los esclavos que se comunican con el Arduino nano a través de los SDA y SCL usados en I2C.

- **Termómetro:**

El sensor ultrasónico se está utilizando para tomar la temperatura solo cuando una persona está a una distancia previamente configurada, cuando la persona se encuentre en la distancia correcta el termómetro se activará para empezar a leer la temperatura.

- **Pantalla de control:**

En la parte de la pantalla de control se mostrará la fecha y la hora actual, además de la temperatura leída y un mensaje indicando si la temperatura es normal o un mensaje de alerta por si la temperatura es alta (esto se mostrará cuando la temperatura ya haya sido leída),

también se mostrará un mensaje de ‘buscando’ cuando no haya una persona presente y un mensaje de ‘leyendo’ cuando la persona se haya puesto frente al termómetro.

- **Portero abre/cierra**

Para la parte del portero se usa el microcontrolador Atmega328p para controlar un motor que simula la puerta de una ciudadela, el cual empezará a girar a la derecha o izquierda de acuerdo con la acción que se desea, abrir o cerrar el portero, el mismo que se activará si uno de los leds (abriendo/cerrando) se encienden y estos se encenderán de acuerdo con el dato que se reciba desde la app creada en appinventor (este dato se recibe a través del Arduino).

## 12. Pruebas de Funcionamiento

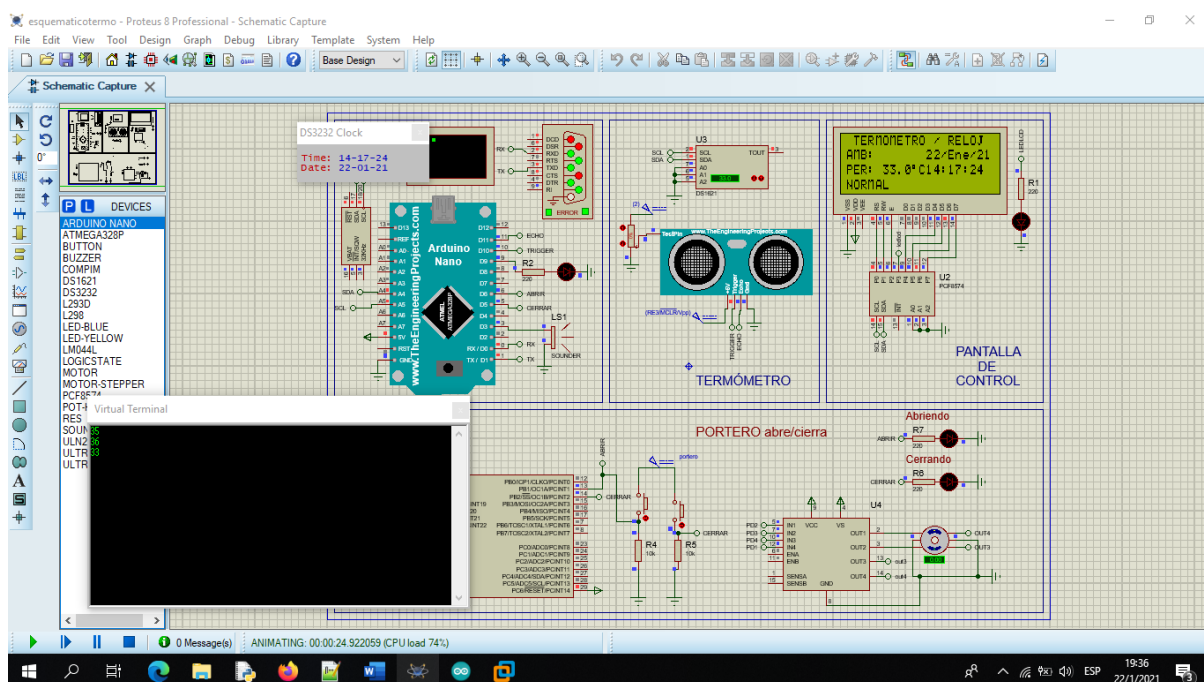


Ilustración 2 lectura de Temperatura

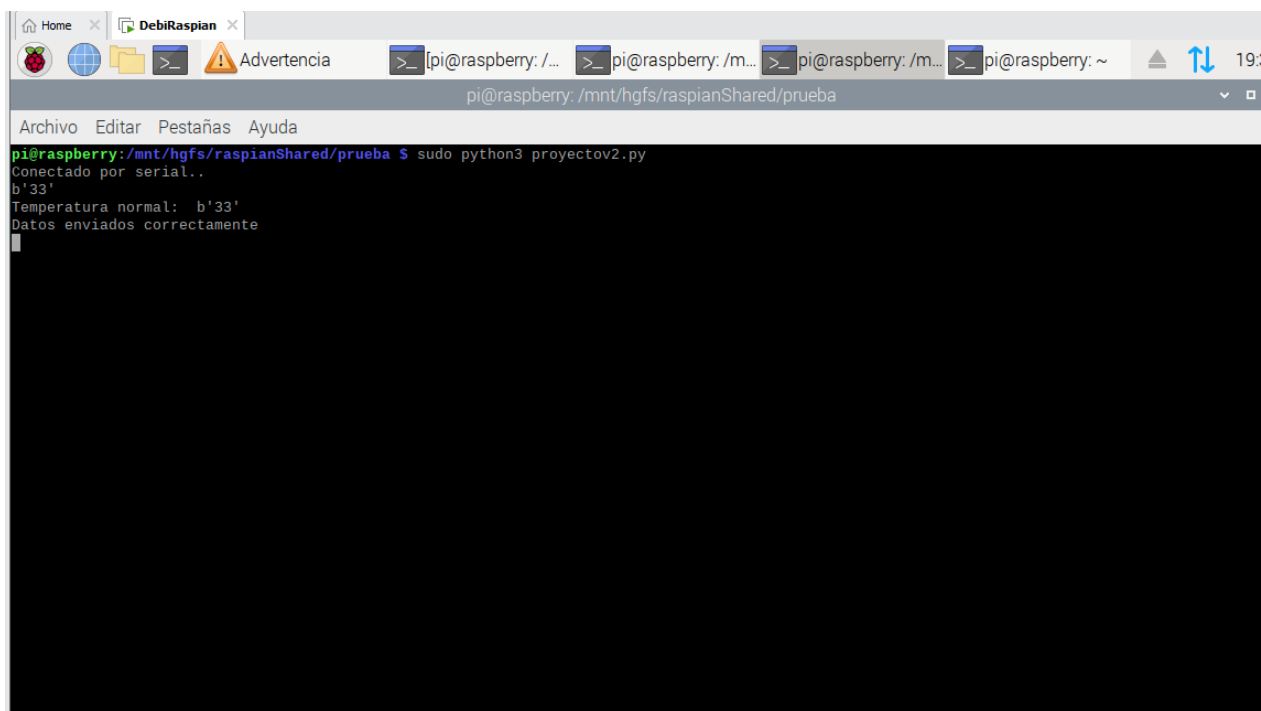


Ilustración 3 Dato de Temperatura enviado a Raspbian

El valor de 33 grados de temperatura enviado desde el Arduino nano a la raspberry pi se muestra en la ilustración 3 y muestra un mensaje de que ese dato fue subido a la nube, para el proyecto se utiliza la plataforma thingspeak para enviar y almacenar de este dato de temperatura.

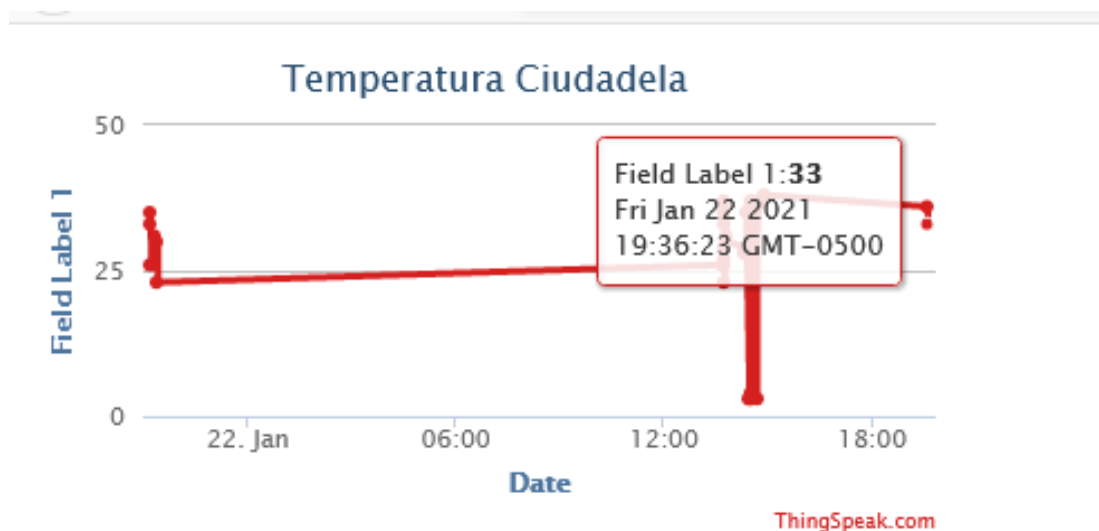


Ilustración 4 Dato de temperatura subido a la nube

En la ilustración 4 se muestra el dato de temperatura ya subido a thingspeak que para el ejemplo es 33 grados.

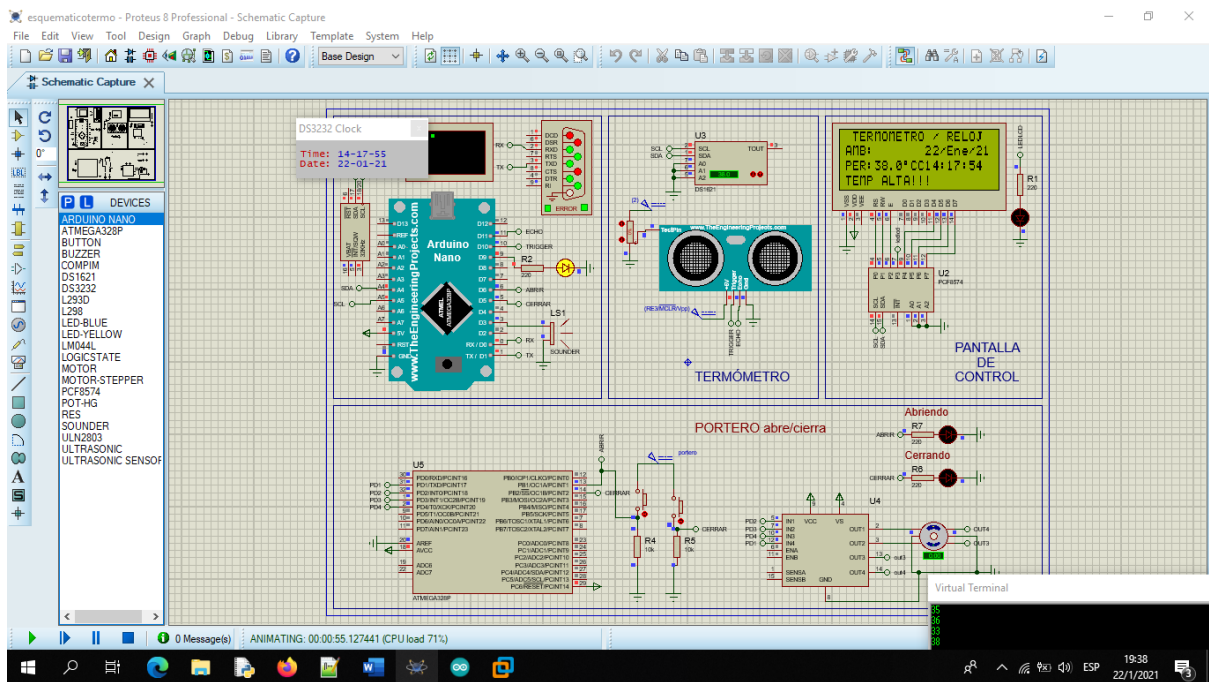


Ilustración 5 Deteccion Temperatura Alta

Home x DebiRaspian x

Advertencia

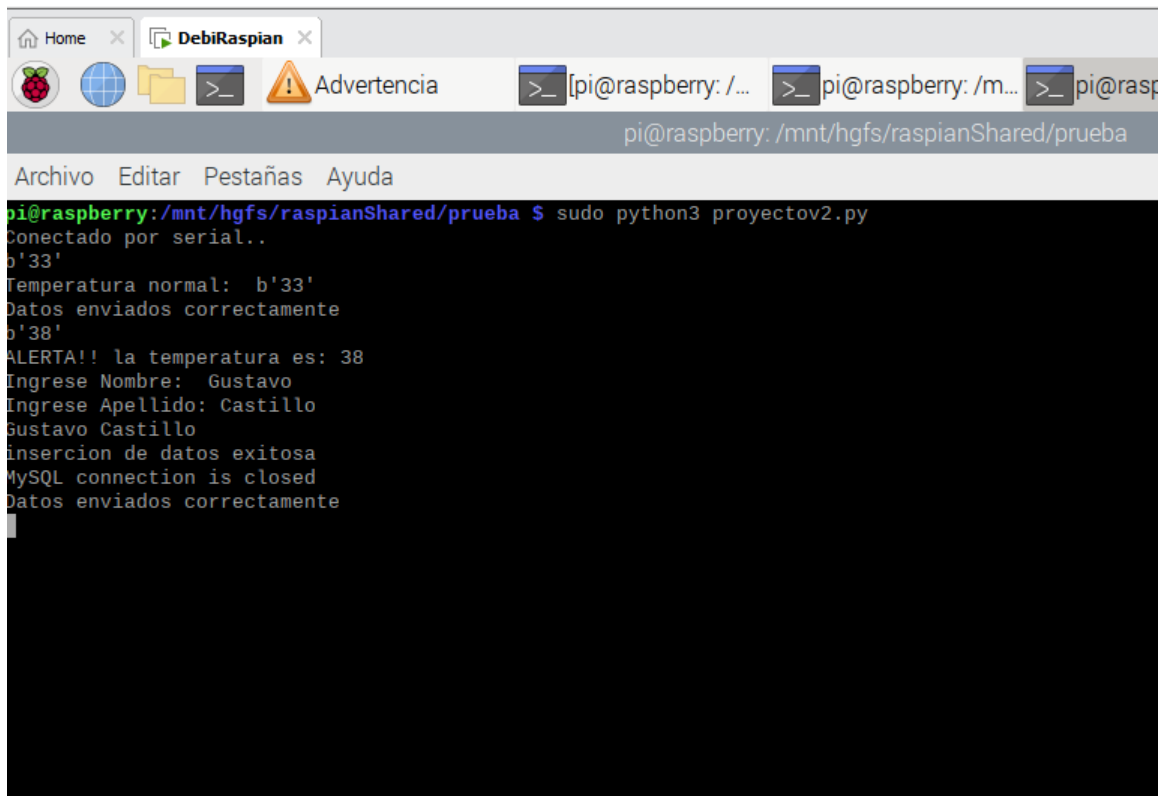
[pi@raspberrypi: /...]

[pi@raspberrypi: /mnt/hgfs/raspianShared/pru]

Archivo Editar Pestañas Ayuda

```
pi@raspberrypi:/mnt/hgfs/raspianShared/prueba $ sudo python3 proyectov2.py
Conectado por serial..
b'33'
Temperatura normal: b'33'
Datos enviados correctamente
b'38'
ALERTA!! la temperatura es: 38
Ingrese Nombre: 
```

Ilustración 6 Se pide datos de la persona con temperatura alta



```
pi@raspberrypi:/mnt/hgfs/raspianShared/prueba $ sudo python3 proyectov2.py
Conectado por serial..
b'33'
Temperatura normal: b'33'
Datos enviados correctamente
b'38'
ALERTA!! la temperatura es: 38
Ingrese Nombre: Gustavo
Ingrese Apellido: Castillo
Gustavo Castillo
Insercion de datos exitosa
MySQL connection is closed
Datos enviados correctamente
```

Ilustración 7 Se ingreso persona a la BD

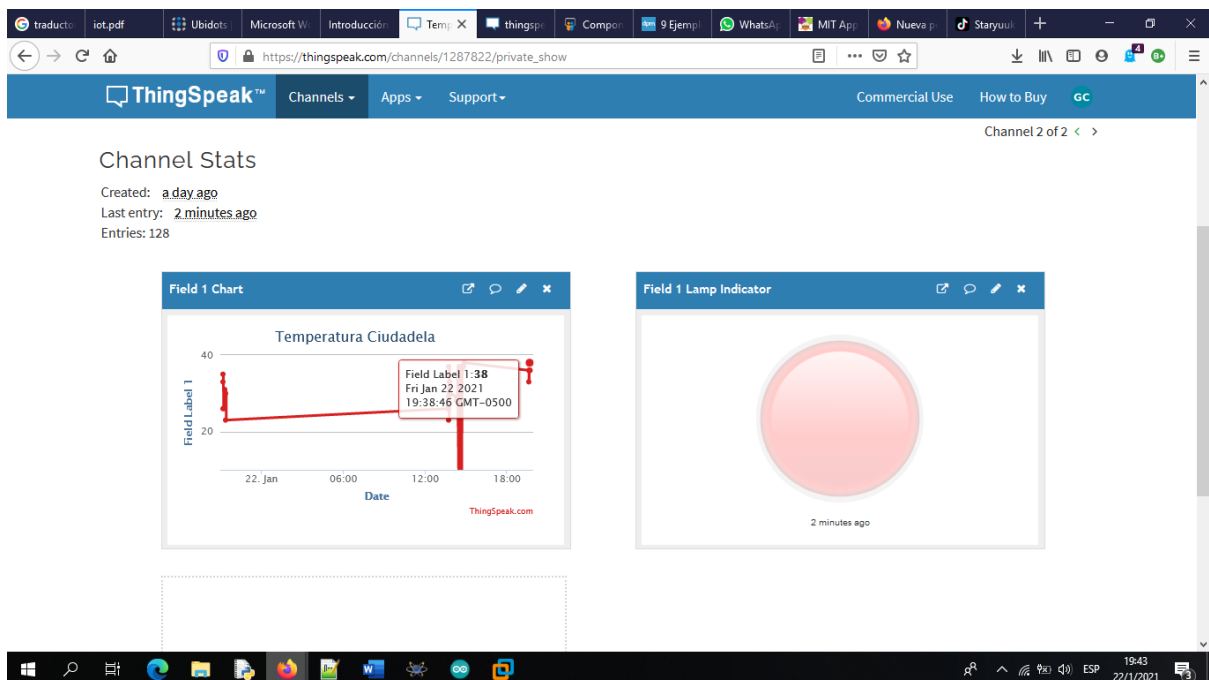


Ilustración 8 Dato de Temperatura de 38 subido a la nube

```
MariaDB [ciudadelaDB]> select*from personas;
```

ID	Nombre	Apellido	Temperatura	fechaHora
1	Gustavo	Castillo	390	2021-01-18 12:00:48
2	Debbie	Donoso	400	2021-01-18 13:54:18
3	Sofia	Mendoza	410	2021-01-18 14:14:50
4	Ricardo	Velez	410	2021-01-18 14:50:58
5	Omar	Cedeño	430	2021-01-18 14:53:33
6	Fernanda	Quevedo	430	2021-01-18 15:18:00
7	Sofia	Vergara	430	2021-01-18 15:45:55
8	Rosa	Solorsano	430	2021-01-18 15:47:34
9	Viviana	Salazar	430	2021-01-18 15:53:39
10	Karla	Pezantes	430	2021-01-18 15:55:31
11	Tania	Velez	430	2021-01-18 15:55:55
12	Andres	Ortiz	430	2021-01-18 15:56:27
13	Belen	Garces	430	2021-01-18 15:56:41
14	Tomas	Pineda	400	2021-01-18 15:57:22
15	Sandra	Lopez	400	2021-01-18 15:57:43
16	Michel	Mendoza	400	2021-01-18 15:58:25
17	Paula	Carbajo	390	2021-01-19 15:57:37
18	Jesenia	Ortiz	380	2021-01-19 16:01:24
19	Andrea	Rodriguez	37	2021-01-22 13:45:50
20	Gustavo	Castillo	37	2021-01-22 14:31:34
21	h	k	37	2021-01-22 14:33:17
22	Nancy	Dueñas	37	2021-01-22 14:53:20
23	Gustavo	Castillo	38	2021-01-22 19:40:52

23 rows in set (0.003 sec)

Ilustración 9 Tabla de base de datos

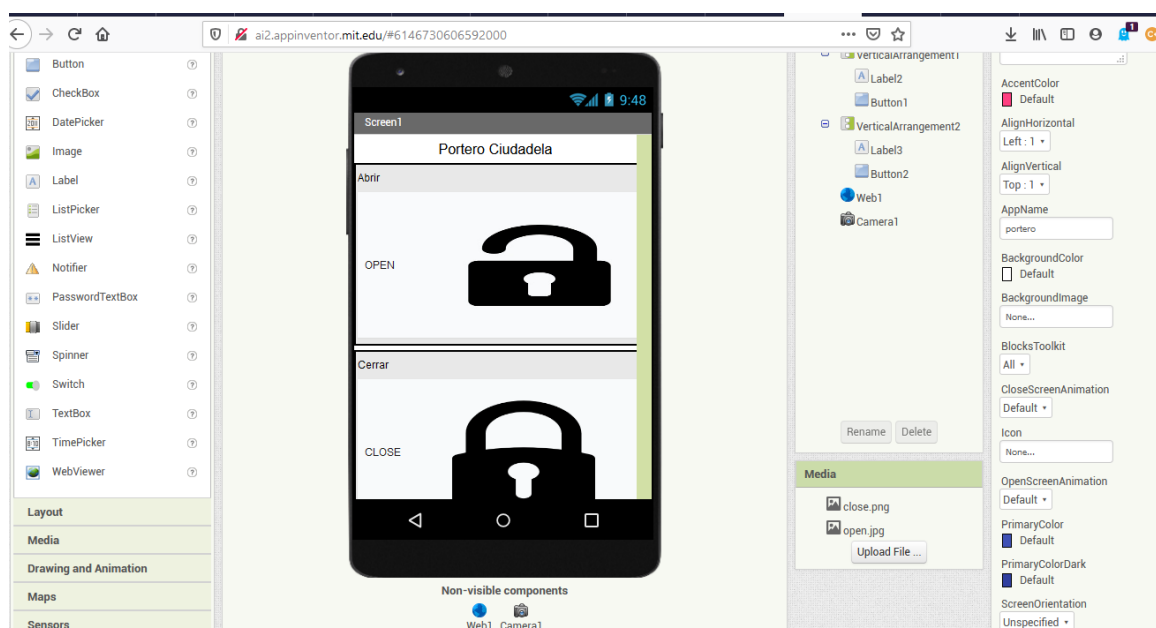


Ilustración 10 aplicación para portero automático



Para la aplicación se utiliza la página <http://ai2.appinventor.mit.edu> donde se puede crear una aplicación y dar orden de abrir o cerrar la portería solo con aplastar un botón en el teléfono Android.

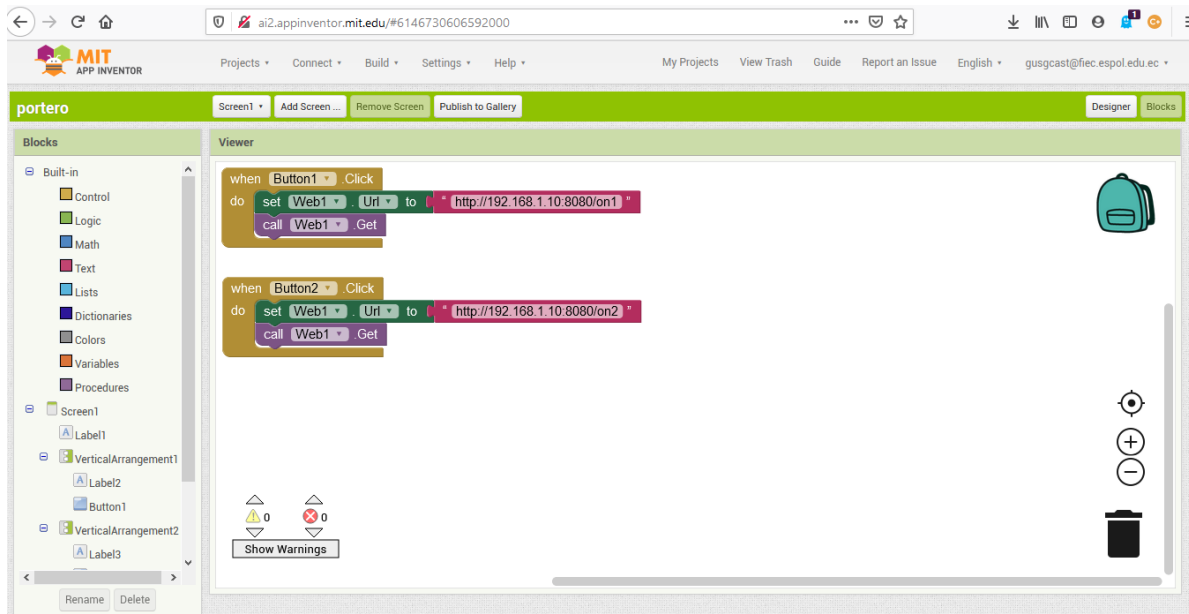


Ilustración 11 Código en bloques de la aplicación

En la ilustración 11 se muestra el código en bloques de dos botones utilizados para encender el motor de la portería el primer botón activa el motor para abrir la portería mientras que el segundo botón activa el motor para cerrar la portería, el comportamiento del motor es que gira para un lado al cerrar y al contrario para abrir.

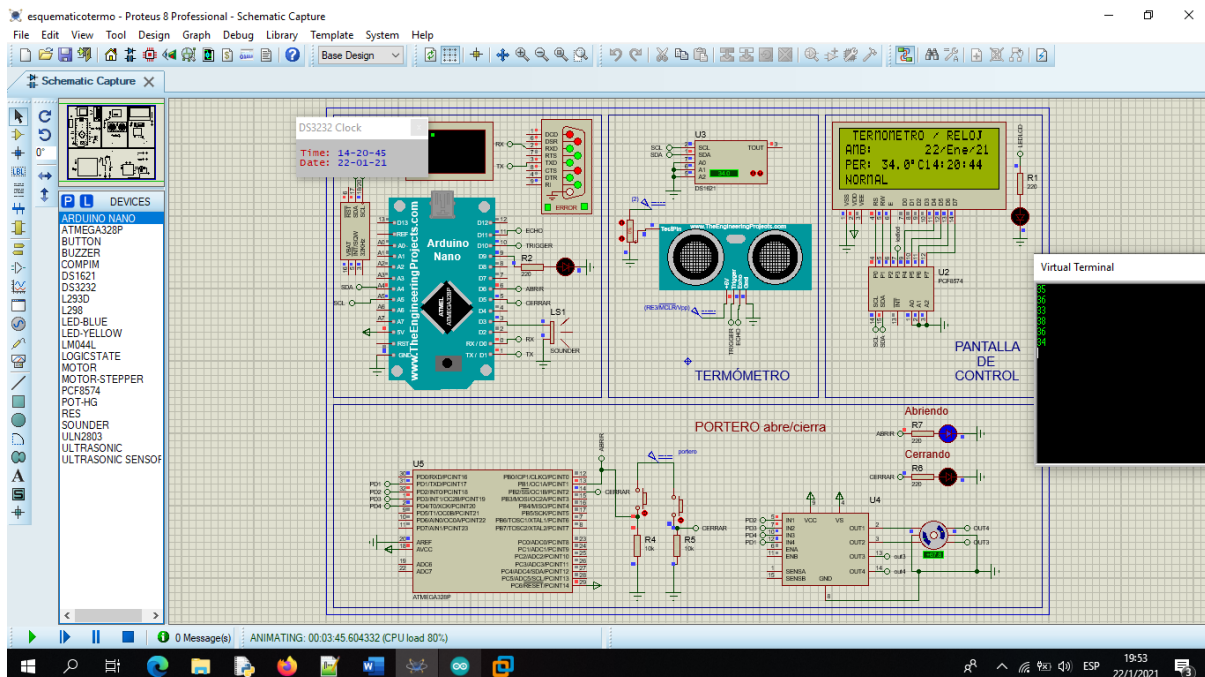


Ilustración 12 Portero Abriendo



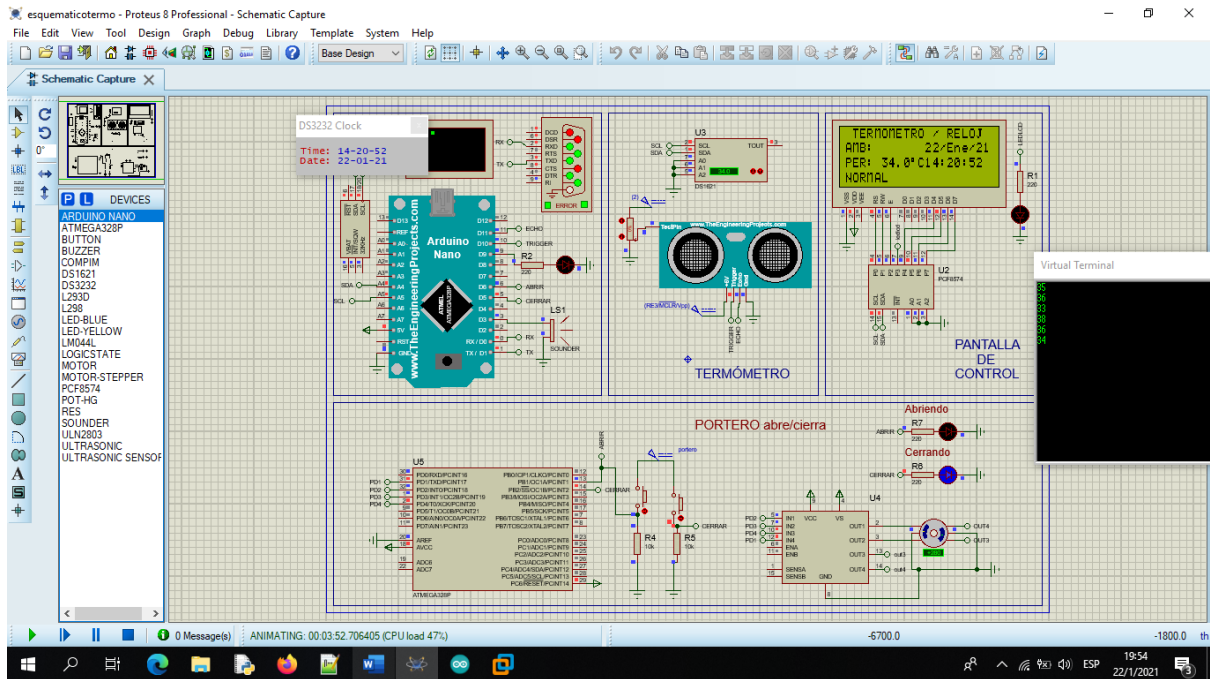


Ilustración 13 Portero Cerrando

```

pi@raspberrypi:~/mnt/hgfs/raspianShared/prueba $ sudo python3 portero.py
conectando a servidor
192.168.1.8 - - [22/Jan/2021 19:52:52] "GET /on1 HTTP/1.1" 200 -
on1
abriendo puerta
192.168.1.8 - - [22/Jan/2021 19:53:08] "GET /on2 HTTP/1.1" 200 -
on2
cerrando puerta
192.168.1.8 - - [22/Jan/2021 19:53:21] "GET /on1 HTTP/1.1" 200 -
on1
abriendo puerta
192.168.1.8 - - [22/Jan/2021 19:53:33] "GET /on2 HTTP/1.1" 200 -
on2
cerrando puerta
192.168.1.8 - - [22/Jan/2021 19:53:51] "GET /on1 HTTP/1.1" 200 -
on1
abriendo puerta
192.168.1.8 - - [22/Jan/2021 19:54:35] "GET /on2 HTTP/1.1" 200 -
on2
cerrando puerta
192.168.1.8 - - [22/Jan/2021 19:55:25] "GET /on1 HTTP/1.1" 200 -
on1
abriendo puerta

```

Ilustración 14 Muestra de conexión para portero

## **VI. ANALISIS DE RESULTADOS**

El sensor de temperatura no estará tomando datos de temperatura todo el tiempo lo cual es muy útil para no tener datos basura que pueden llegar a causar ruido de la información de esta forma según como ocurran los eventos en el día a día en la portería se podrá observar en las gráficas de thingspeak que durante las horas del día pueden llegar varios puntos de temperatura que se han subido y en la noche la concurrencia debería disminuir además que se puede observar por ejemplo en un mes como ha sido el tráfico de las personas que han ingresado a la ciudadela.

## **VII. CONCLUSIONES**

- El diseño de sistema de monitoreo permite obtener la temperatura que tiene una persona y determinar si esta persona puede llegar a tener fiebre lo cual es importante para permitir el acceso a la residencia ya que la fiebre puede ser síntoma de un posible contagio de covid-19 y para evitar más contagios a las personas en la ciudadela se puede cuestionar el ingreso de esta.
- Se puede saber exactamente la fecha y la hora en la que una persona se registró con una alta temperatura y esa información es importante para determinar un posible desarrollo de la enfermedad para el contexto de los días en que fue registrado.
- Los dispositivos como el Arduino nano, y la raspberry pi actúan como transmisor y receptor de datos ya que por un lado el Arduino nano puede enviar datos de los sensores a la raspberry pi y la raspberry pi envía datos para las instrucciones que debe ejecutar el atmega328p, además de que la raspberry pi puede enlazarse con distintos servidores para almacenar la información en la nube y poder observar los datos desde cualquier lugar.

## **VIII. RECOMENDACIONES**

- Para crear una base de datos en la nube se recomienda comprar un plan en la cuenta de clever cloud, ya que si se crea la base con una de prueba solo se podrá ingresar 5 veces a la misma.
- Si se desea replicar este proyecto de manera virtual es necesario contar con memoria suficiente, puesto que los programas utilizados deben estar abiertos al mismo tiempo para poder observar su funcionamiento.
- Para replicar el proyecto de manera física es recomendable modificar ciertos aspectos de los códigos, además la comunicación del Arduino con la Raspberry Pi sería de manera serial (USB) y se deberían agregar periféricos adicionales para la Raspberry, así como un teclado.
- Para mayor seguridad y control de la ciudadela se podría implementar una cámara, además contar con una base de datos con las imágenes de los residentes con sus respectivos datos de manera que al ingresar a la ciudadela se detecte la cara de las personas y si coincide con una de las de la base de datos se mostrará la información en la pantalla.

- Si se desea realizar la comunicación serial del Arduino con la Raspberry Pi de manera virtual se debe tener un programa para simular el par virtual, se recomienda usar el programa de configure virtual Serial Port Driver.
- La raspberry pi debe estar enlazada a la misma red local en la que están enlazados los dispositivos que utilizan la aplicación de portero automático.

## IX. BIBLIOGRAFÍA

Administrador. (28 de octubre de 2018). *hetpro-store*. Obtenido de hetpro-store:  
<https://hetpro-store.com/TUTORIALES/i2c/>

De Girodmedical, M. (19 de Febrero de 2020). Obtenido de  
[https://www.girodmedical.es/blog\\_es/como-funciona-un-termometro-infrarrojo/](https://www.girodmedical.es/blog_es/como-funciona-un-termometro-infrarrojo/)

geekfactory. (s.f.). *geekfactory*. Obtenido de geekfactory:  
<https://www.geekfactory.mx/tienda/motores-actuadores-servos-y-accesorios>

Hernández, L. (2018). *programar facil*. Obtenido de <https://programarfácil.com/blog/arduino-blog/termometro-infrarrojo-con-arduino-mlx90614/>

Integrated, M. (2015). Obtenido de <https://pdfserv.maximintegrated.com/en/ds/DS1621.pdf>

luisllamas. (s.f.). *luisllamas.es*.

Manrique. (23 de Diciembre de 2016). *Servomotores: apertura automática de puerta de garaje*. Obtenido de <http://fpkanarias.blogspot.com/2016/12/servomotores-apertura-automatica-de.html>

*matic-port*. (1 de Octubre de 2018). Obtenido de <https://www.puertasautomaticasmatic-port.com/funcionamiento-de-las-puertas-automaticas/#:~:text=Las%20puertas%20autom%C3%A1ticas%20tienen%20un,que%20debe%20abrirse%20o%20cerrarse.&text=Estas%20est%C3%A1n%20conectadas%20por%20cables,y%20cierre%20de%20las%2>

SERMER. (25 de Mayo de 2018). Obtenido de <https://www.sermer.com.ni/las-6-ventajas-de-los-termometros-infrarrojos/>

## X. ANEXOS

### 13. Código Arduino

```
//Librerias
#include <Wire.h> //comms
#include <LiquidCrystal_I2C.h> //Pantalla
#include <DS3231.h> //Reloj
#include <SR04.h> //Ultrasonico
#include "pitches.h" //Buzzer

#define DS1621_ADDRESS 0x48

//Definicion de Pines
#define Echo 11 //Echo del Ultrasonico
#define Trig 10 //Trig del Ultrasonico
#define LP 9 //Salida LED
#define buzz 3
#define abre 1 //para portero
#define cierra 2 //para portero

//Inicializar
//Adafruit_MLX90614 mlx = Adafruit_MLX90614(); //Sensor Termico
LiquidCrystal_I2C lcd(0x20,20,4);
DS3231 clock; //Reloj
RTCDatetime dt; //Fecha
SR04 sr04=SR04(Echo,Trig); //Ultrasonico

//Variables
int Espera1=500; //Espera en el loop
float sinVal;
int toneVal;
const int ledRasp=6;
const int ledD5=5;

//Fecha
String fDia;
String fMes; //Mes en texto
String fAno;
String fHora;
String fMin;
String fSeg;
String fTime;

//Distancia
int Dist; //Distancia del ultrasonico
int DistMin=10; //Distancia minima para detectar al sujeto (mm)
int Presente=0; //Si hay alguien frente al Termometro
```

```

int Espera=3000; //Tiempo de espera para verificar sujeto
unsigned long Tiempo=0; //Tiempo que lleva detectado para Millis
int Ahora=0; //Millis en el momento que se inicia

//Temperatura
//float TempObj; //Temperatura del sujeto
float TempMax=370; //Temperatura maxima permitida
int TpoAlarma=200; //Tiempo de Alarma por alta temperatura
int tempC;
int tempAnte=0;

void setup()
{
  pinMode(buzz, OUTPUT);

  pinMode(LP,OUTPUT); //LED
  pinMode(ledRasp,OUTPUT);
  pinMode(ledD5,OUTPUT);
  //mlx.begin(); //Termico
  Serial.begin(9600);

  clock.begin(); //Reloj
  clock.setDateTime(2021,1,22,14,17,0); //Quitar comentario y ajustar a la hora deseada en primer
  run.

  digitalWrite(LP,HIGH);
  delay(1000);
  digitalWrite(LP,LOW);

  //Mensaje inicial
  lcd.init(); //inicio lcd
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(" TERMOMETRO / RELOJ");
  lcd.setCursor(0,1);
  lcd.print("AMB: ");
  lcd.setCursor(0,2);
  lcd.print("PER: ");

  Wire.begin();      // join i2c bus
  // REGISTRO DE CONTROL
  //=====
  Wire.beginTransmission(DS1621_ADDRESS); //Condicion de INICIO y direcciono el SLAVE
  Wire.write(0xAC); //Mando comando para escribir en el registro de control
  Wire.write(0x02);
  Wire.endTransmission();
  //Configuro para una adquisicion continua

```

```

//=====
// REGISTRO ALARMA TERMOSTATO TH-TL
//=====
Wire.beginTransmission(DS1621_ADDRESS); //Condicion de INICIO y direcciono el SLAVE
Wire.write(0xA1); //Mando comando para escribir en el registro TH
Wire.write(0x1D); //Configuro T maxima del termostato
Wire.write(0x00);
Wire.endTransmission();
delay(500);
Wire.beginTransmission(DS1621_ADDRESS); //Condicion de INICIO y direcciono el SLAVE
Wire.write(0xA2); //Mando comando para escribir en el registro TL
Wire.write(0x19); //Configuro T minima del termostato
Wire.write(0x00);
Wire.endTransmission();
//=====
// REGISTRO DE COMIENZO DE CONVERSION
//=====
Wire.beginTransmission(DS1621_ADDRESS); //CONDicion de INICIO REPETIDA
Wire.write(0xEE); //Mando comando para que de comienzo la conversion
Wire.endTransmission(); //Condicion de STOP // no hace falta utilizar esta funcion

}

// variables
char c_buffer[8], f_buffer[8];
void loop()
{
    delay(100);
    Wire.beginTransmission(DS1621_ADDRESS); //Condicion de INICIO
    Wire.write(0xAA); //Mando comando para leer la ultima conversion
    Wire.endTransmission(); // STOP
    Wire.requestFrom(0x48, 1); //Realizo una peticion de lectura al SLAVE y de un Byte
    //tempC = Wire.read(); //Guardo el resultado
    tempC = get_temperature();
    int datoRasp=Serial.read()-'0';

    if(datoRasp==abre){
        digitalWrite(ledRasp,HIGH);
    }
    else{
        digitalWrite(ledRasp,LOW);
    }
    if(datoRasp==cierra){
        digitalWrite(ledD5,HIGH);
    }
    else{

```

```

    digitalWrite(ledD5,LOW);
}

//int tempWhole = tempC >> 8;
//int tempFrac = (tempC & 0xFF) / 256;
//float temp = tempWhole + tempFrac;

//Distancia
Dist=sr04.Distance();
if(Dist>DistMin) //No hay nadie
{
    Presente=0;
    Tiempo=millis();
}
if(Dist<=DistMin && Presente==0) //Llego alguien, tomemos el tiempo
{
    Presente=1;
    Tiempo=millis();
}

if(Presente==1)
{
    if(millis()-Tiempo>Espera) //Se completo el tiempo
    {
        Presente=2;
    }
}

//Actualizar pantalla
lcd.setCursor(12,3);

//Temperaturas
lcd.setCursor(4,2);
lcd.print(c_buffer);
lcd.setCursor(8,2);
//lcd.print("C");

//TempObj=mlx.readObjectTempC();
switch(Presente)
{
    case 0: //No hay nadie
        lcd.setCursor(4,2);
        lcd.print("---c");
        lcd.setCursor(0,3);
        lcd.print("BUSCANDO... ");
        break;

    case 1: //Llego alguien
        lcd.setCursor(4,2);
        lcd.print("---c");

```

```

lcd.setCursor(0,3);
lcd.print("LEYENDO...");
//tone(12,NOTE_C5,TpoAlarma);
break;

case 2: //Se Completo el tiempo
lcd.setCursor(4,2);
lcd.print(c_buffer);
lcd.setCursor(8,2);
//lcd.print("C");

if(tempC > TempMax) { // if temperature < 0 °C
    tempC = abs(tempC); // absolute value
    sprintf(c_buffer, "%02u.%1u°C", tempC / 10, tempC % 10, 223);
    //tone(12,NOTE_G5,TpoAlarma);
    lcd.setCursor(0,3);
    lcd.print("TEMP ALTA!!!");
    digitalWrite(LP,HIGH);
    delay(TpoAlarma);
    digitalWrite(LP,LOW);
    //Serial.println(tempC);
    tempC=tempC/10;
    if(tempAnte!=tempC){
        Serial.println(tempC);
        tempAnte=tempC;
    }
    for (int x=0; x<180; x++) {
// pasamos de grados a radianes
        sinVal = (sin(x*(3.1412/180)));
// Generamos el tono, o mejor dicho la frecuencia
        toneVal = 2000+(int(sinVal*1000));
        tone(3, toneVal,2);
    }

}
else {
    //if (tempC > 128) // if temperature >= 100.0 °C
    //sprintf(c_buffer, "%03u.%1u°C", c_temp / 10, c_temp % 10, 223);
    //else
    sprintf(c_buffer, " %02u.%1u°C", tempC / 10, tempC % 10, 223);
    lcd.setCursor(0,3);
    lcd.print("NORMAL    ");

    tempC=tempC/10;
    if(tempAnte!=tempC){
        Serial.println(tempC);
        tempAnte=tempC;
    }

}

}
break;

```



```

}

//Hora y Fecha
dt=clock.getDateTime();
fDia=String(dt.day);
if(dt.day<10)fDia="0"+fDia; //Para mantener en dos digitos

fAno=String(dt.year-2000);

lcd.setCursor(11,1);
lcd.print(fDia+"/"+ Mes()+"/"+fAno); //Uso de la funcion Mes()

fHora=String(dt.hour);
if(dt.hour<10)fHora="0"+fHora; //Para mantener en dos digitos

fMin=String(dt.minute);
if(dt.minute<10)fMin="0"+fMin; //Para mantener en dos digitos

fSeg=String(dt.second);
if(dt.second<10)fSeg="0"+fSeg; //Para mantener en dos digitos

fTime=fHora+":" + fMin + ":" + fSeg;
lcd.setCursor(11,2);
lcd.print(fTime);

delay(Espera1);

}

////////////////////////////////////
////////////////////////////////////
/// F U N C I O N E S
////////////////////////////////////
////////////////////////////////////

String Mes()
{
  //Serial.println(dt.month);
  switch (dt.month)
  {
    case 1:
      fMes="Ene";
      break;

    case 2:
      fMes="Feb";
      break;
  }
}

```

```

    case 3:
    fMes="Mar";
    break;

    case 4:
    fMes="Abr";
    break;

    case 5:
    fMes="May";
    break;

    case 6:
    fMes="Jun";
    break;

    case 7:
    fMes="Jul";
    break;

    case 8:
    fMes="Ago";
    break;

    case 9:
    fMes="Sep";
    break;

    case 10:
    fMes="Oct";
    break;

    case 11:
    fMes="Nov";
    break;

    case 12:
    fMes="Dic";
    break;
}

return fMes;
}

int16_t get_temperature() {
    Wire.beginTransmission(DS1621_ADDRESS); // connect to DS1621 (send DS1621 address)
    Wire.write(0xAA);                       // read temperature command
    Wire.endTransmission(false);            // send repeated start condition
    Wire.requestFrom(DS1621_ADDRESS, 2);    // request 2 bytes from DS1621 and release I2C bus
    at end of reading

```

```

uint8_t t_msb = Wire.read();    // read temperature MSB register
uint8_t t_lsb = Wire.read();    // read temperature LSB register

// calculate full temperature (raw value)
int16_t raw_t = (int8_t)t_msb << 1 | t_lsb >> 7;
// convert raw temperature value to tenths °C
raw_t = raw_t * 10/2;
return raw_t;
}

```

## 14. Código Raspbian

```

import time
import serial
import mysql.connector
from mysql.connector import Error
import os
import io
import requests
import json
temperatura=0

def insertVariablesIntoTable(nombre,apellido,temp):
    try:
        connection = mysql.connector.connect(host='localhost',
                                             database='ciudadelaDB',
                                             user='gustavoCastillo',
                                             password='123')
        cursor = connection.cursor()
        mySql_insert_query = """INSERT INTO personas (Nombre,Apellido,Temperatura)
                                VALUES (%s, %s, %s) """

        recordTuple = (nombre,apellido,temp)
        cursor.execute(mySql_insert_query, recordTuple)
        connection.commit()
        print("insercion de datos exitosa")

    except mysql.connector.Error as error:
        print("Failed to insert into MySQL table {}".format(error))

    finally:
        if (connection.is_connected()):
            cursor.close()
            connection.close()
            print("MySQL connection is closed")

arduino = serial.Serial(
    port='/dev/ttyS1',
    baudrate = 9600,

```

```

    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=1
)
print("Conectado por serial..")
arduino.flush()

try:
    while(1):
        if(arduino.in_waiting>0):
            temperatura=arduino.readline(2).strip()
            print(temperatura)
            enviar =
requests.get("https://api.thingspeak.com/update?api_key=WCLYG0G6ALZJXKK0&field1="+str(temperatura.decode()))
            if float(temperatura)>=37:
                print("ALERTA!! la temperatura es:",int(temperatura))
                nombre=input("Ingrese Nombre: ")
                apellido=input("Ingrese Apellido: ")
                print(nombre,apellido)
                insertVariablesIntoTable(nombre,apellido,temperatura)

                temperatura=arduino.readline(2).strip()
                #print("Tablas de 'ciudadela':")
                #for fila in filas:
                #    print(fila)
            else:
                print("Temperatura normal: ",temperatura)
                temperatura=arduino.readline(2).strip()

            if enviar.status_code == requests.codes.ok:
                if enviar.text != '0':
                    print("Datos enviados correctamente")
                else:
                    print("Tiempo de espera insuficiente (>15seg)")
            else:
                print("Error en el request: ",enviar.status_code)

        recibir=requests.get("https://api.thingspeak.com/channels/7NUASKBQJLYQCE67/feeds.json")
        jsonString=json.dumps(recibir.json(),indent=2)
        time.sleep(15)
except KeyboardInterrupt, SystemExit:
    print()
    print("hasta luego")
    arduino.close()

```

## 15. Código Portero para APP en Raspbian

```
import serial
import time
from http.server import BaseHTTPRequestHandler, HTTPServer

Request = None
arduino = serial.Serial(
    port='/dev/ttyS1',
    baudrate = 9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=1
)
time.sleep(2)
class RequestHandler_httpd(BaseHTTPRequestHandler):
    def do_GET(self):
        global Request
        messagetosend = bytes('Solicitando conexion','utf')
        self.send_response(200)
        self.send_header('Content-Type', 'text/plain')
        self.send_header('Content-Length', len(messagetosend))
        self.end_headers()
        self.wfile.write(messagetosend)
        Request = self.requestline
        Request = Request[5 : int(len(Request)-9)]
        print(Request)
        if Request == 'on1':
            print('abriendo puerta')
            arduino.write(str(1).encode('utf-8'))
        if Request == 'on2':
            print('cerrando puerta')
            arduino.write(str(2).encode('utf-8'))

server_address_httpd = ('192.168.1.10',8080)
httpd = HTTPServer(server_address_httpd, RequestHandler_httpd)
print('conectando a servidor')
httpd.serve_forever()
```

## 16. Código Atmega328p

```
#define F_CPU 8000000
#include <avr/io.h>
#include <util/delay.h>
uint8_t valor1;
```

```

uint8_t valor2=2;
uint8_t valida(uint8_t t){
    return t;
}
int main(void)
{
    /* Replace with your application code */
    DDRD=0b00011111;
    DDRB &= ~(1<<DDB1);
    DDRB &= ~(1<<DDB2);
    PORTD &= ~(1<<PD2);
    PORTD &= ~(1<<PD3);
    PORTD &= ~(1<<PD4);
    PORTD &= ~(1<<PD5);

    while (1)
    {

        if((PINB & (1<<PB1)) && (valor2==2)){
            valor2=valida(1);
            valor1=valida(2);
            //izquierda
            PORTD |= (1<<PD2);
            PORTD |= (1<<PD3);
            PORTD &= ~(1<<PD4);
            PORTD &= ~(1<<PD1);
            _delay_ms(20);
            //arriba
            PORTD |= (1<<PD2);
            PORTD &= ~(1<<PD3);
            PORTD &= ~(1<<PD4);
            PORTD |= (1<<PD1);
            _delay_ms(20);
            //derecha
            PORTD &= ~(1<<PD2);
            PORTD &= ~(1<<PD3);
            PORTD |= (1<<PD4);
            PORTD |= (1<<PD1);
            _delay_ms(20);
            //abajo
            PORTD &= ~(1<<PD2);
            PORTD |= (1<<PD3);
            PORTD |= (1<<PD4);
            PORTD &= ~(1<<PD1);
        }
        if((PINB & (1<<PB2)) && (valor2==1)){
            valor2=valida(2);

            //derecha
            PORTD &= ~(1<<PD2);
            PORTD &= ~(1<<PD3);

```

```
        PORTD |= (1<<PD4);
        PORTD |= (1<<PD1);
        _delay_ms(20);
        //arriba
        PORTD |= (1<<PD2);
        PORTD &=~(1<<PD3);
        PORTD &=~(1<<PD4);
        PORTD |= (1<<PD1);
        //izquierda
        _delay_ms(20);
        PORTD |= (1<<PD2);
        PORTD |= (1<<PD3);
        PORTD &=~(1<<PD4);
        PORTD &=~(1<<PD1);
        _delay_ms(20);
        //abajo
        PORTD &= ~(1<<PD2);
        PORTD |= (1<<PD3);
        PORTD |= (1<<PD4);
        PORTD &= ~(1<<PD1);

    }

}
```