Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Control de riego distribuido para pequeñas explotaciones

Autor:

Galán Rodríguez, Raúl

Tutor(es):

Arranz Gimón, Ángel Eugenio

Departamento de Tecnología Electrónica

Valladolid, junio de 2024.

RESUMEN

En este trabajo se desarrollará la implementación de un sistema de riego automático distribuido para hacer uso de este en pequeñas explotaciones basado en la tarjeta Arduino UNO Rev3. El riego se activará en función de las necesidades del ambiente o de las condiciones requeridas por el usuario.

Para poder desarrollar este proyecto es necesario contar con una serie de sensores que recojan información del terreno y unos pequeños actuadores que permitan tener un control sobre el riego en todo momento. A partir de la información que recojan los sensores, el sistema actuará en consecuencia. El proyecto contará con una aplicación móvil que permita controlar y visualizar de forma remota el riego.

Para este proyecto se cuenta con una maqueta física donde se integrarán componentes básicos que cualquier instalación debería tener para su correcto funcionamiento. La maqueta solo representará la parte principal de la instalación para explicar y comprobar su funcionamiento. Sin embargo al tratarse de una maqueta, los componentes no serán los idóneos para lograr la máxima eficiencia pues se utilizan componentes con unas características limitadas.

PALABRAS CLAVE

Arduino, control de riego, sensores, comunicación I2C, aplicación móvil.

ABSTRACT

In this work, the implementation of a distributed automatic irrigation system will be developed to make use of it in small farms based on the Arduino UNO Rev3 card. Irrigation system will be triggered based on the needs of the environment or the conditions required by the user.

To develop this project, it is necessary to have series of sensors that collect information from the terrain and small actuators that allow control over irrigation. Based on the information collected by the sensors, the system will act accordingly. The project will have a mobile application that allows remote control and visualization of irrigation.

For this project there is a physical model where basic components that any installation should have for its correct operation will be integrated. The model will only represent the main part of the installation to explain and check its operation. However, as it is a mock-up, the components will not be the ideal ones to achieve maximum efficiency, but components with limited characteristics will be used.

KEYWORDS

Arduino, automated irrigation control, sensor, I2C communication, mobile application.

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN

EPÍGRAFE 1.1. JUSTIFICACIÓN DEL PROYECTO

La justificación de este proyecto es controlar el gasto de agua en todo momento en función de las necesidades requeridas por el terreno. Para un óptimo control de gasto de agua de regadío es necesario conocer la humedad del terreno y temperatura del ambiente. Si se dispone de un sólo sensor de humedad, el control del riego derivado de esta información solo es preciso en la zona donde se ha medido. Para solventar en parte esta situación, se pretende diseñar una red de sensores de humedad y temperatura que transmitan su información a un controlador que, mediante la ponderación de los datos recibidos, decida activar el riego o detenerlo. Para la transmisión de la información de los sensores puede utilizarse cableado y protocolo de comunicación IoT. El propietario de la explotación podrá conectarse vía Bluetooth con el controlador, para tener un control más accesible y rápido.

EPÍGRAFE 1.2. OBJETIVOS Y ALCANCE DEL PROYECTO

El objetivo principal de este proyecto es crear un pequeño control de riego para pequeñas explotaciones, como pueden ser, macetas, jardines o algún pequeño huerto donde se deben tener en cuenta los siguientes aspectos:

- Conocer las condiciones del terreno.
- Transmisión de la información recogida por los sensores a un dispositivo central.
- Control de la cantidad de agua que necesita la instalación en cada momento.
- Parámetros de humedad, temperatura, horas de riego programables.
- Conexión vía telemática para una configuración y control remoto.
- Versatilidad.

Como ya se ha comentado en la introducción el desarrollo de este proyecto trata de construir una pequeña maqueta controlada mediante una aplicación móvil que simule a pequeña escala como podría funcionar una instalación de mayor tamaño con componentes de mejor calidad, funcionalidad y alcance.

CAPÍTULO 2. MARCO TEÓRICO

EPÍGRAFE 2.1. ANTECEDENTES

El riego tiene una historia que se remonta alrededor del año 6.000 a. C en Egipto y posteriormente en Mesopotamia, donde se empezaron a construir canales y diques para poder aprovechar las grandes crecidas causadas por los ríos Nilo, Tigris y Éufrates. Mediante este sistema mejoraron la cosecha, así como también permitió mejorar la cría de ganado.

Alrededor del año 3000 a. C se llevó a cabo el primer gran proyecto de irrigación en el que las principales herramientas de las que se disponían eras los canales y las presas, destinadas a poder almacenar grandes cantidades de agua para su posterior uso. Con este sistema se construyó el primer lago artificial llamado "Moeris".[2]

Otro sistema de riego creado hace miles de años son los 'baolis' de India, se parecen a un tipo de pirámide inversa, muy anchas en la superficie y estrechas en el fondo, que cuenta con escalones y pasarelas para poder moverse entre diferentes niveles. Fueron creados para poder almacenar el agua de las numerosas lluvias torrenciales, para después poder afrontar los grandes periodos de sequía.[4]

Figura 2.1. Baolis de la India [4]

Los canales y acueductos romanos son sistemas de abastecimiento hidráulico, en el que su objetivo era transportar agua desde localizaciones alejadas de la civilización, ya fuera para consumo humano o sistemas de riego para cultivos. Lo normal es que estos canales fueran subterráneos, pero cuando había que atravesar lugares con desniveles muy pronunciados se construían acueductos.[6]

Figura 2.2. Acueducto romano [6]

El agua ha sido y es fundamental en la vida diaria para todos los seres vivos, por eso hemos ido creando técnicas que nos faciliten el acceso a su uso. Entre las técnicas más conocidas de la antigüedad podemos destacar las comentadas anteriormente, aunque sin duda existen muchas más.

EPÍGRAFE 2.2. DEFINICIÓN TEÓRICA Y ACTUALIDAD

Podemos definir el riego como la tecnología que busca poder transportar el agua hacía las plantas, jardines, macetas, cultivos… para que se puedan reproducir y alimentar correctamente.

En la actualidad, podemos ver como esta tecnología no ha parado de evolucionar y de buscar mejoras y alternativas para poder sacar un mejor rendimiento y productividad al campo.

Existen multitud de tipos de riego y estos se pueden catalogar en dos grupos bastante diferenciados como son el riego por superficie o de gravedad y el riego presurizado.

- El riego por superficie o de gravedad está presente sobre todo en áreas cercanas a embalses, ríos o canales desde donde se pueden derivar pequeñas tuberías que desembocan en los surcos del terreno. Presenta pocas ventajas frente a la gran cantidad de inconvenientes que tiene como, por ejemplo, el terreno debe de ser completamente llano para que el agua se distribuya uniformemente, ya que si no unas zonas quedarían más inundadas de agua que otras y la producción no sería la misma, es necesaria gastar una cantidad de agua muy elevada para que toda la zona queda inundada por una fina capa de agua, hay que tener muy en cuenta el tipo de terreno que se está explotando y saber qué tipo de filtraciones puede tener, ya que si es muy poroso una gran parte de la cantidad de agua que utilicemos no se va a poder aprovechar ya que se va a filtrar, y al contrario,

si es muy impermeable es posible que se inunde el cultivo por un exceso de agua y se pierda la cosecha. [10]

Figura 2.3. Riego por inundación o de gravedad [11]

- El riego presurizado es el más común en la actualidad, está presente en diferentes técnicas pero el concepto es el mismo. Podemos distinguir los siguientes:
  - Riego por aspersión: nos permite aplicar el agua en forma de lluvia sobre el cultivo. Los inconvenientes que se deben de tener más en cuenta a la hora de utilizar este tipo de riego de aspersión son que debe de utilizarse en áreas en las que el viento no sea elevado, ya que si no, no sería eficiente y además requiere de mucha energía eléctrica al necesitar una elevada presión de funcionamiento. A pesar de estos dos inconvenientes, tiene una gran cantidad de ventajas ya que es uno de los sistemas más eficientes, cubre grandes áreas de terreno y se puede automatizar. Dentro de este tipo de riego, hay diferentes formas de instalación, las más comunes son:

- Tuberías fijas mediante aspersores, ya sean permanentes o temporales.

Figura 2.4. Riego por aspersión mediante aspersores [11]

- De desplazamiento continuo como los pívots o ala sobre carros.

Figura 2.5. Riego por aspersión mediante pivot o lateral de avance frontal [11]

- Riego por microaspersión: el funcionamiento es exactamente igual que el riego por aspersión sin embargo, la ventaja que tiene es que necesita una presión mucho más baja ya que las gotas de agua que pulveriza son mucho más finas.

Figura 2.6. Riego por microaspersión [12]

- Riego por goteo o localizado: consiste en la aplicación continua de agua en forma de gotas en un lugar específico del cultivo, por lo que se consigue formar un bulbo húmedo bajo cada gotero que permite un mejor desarrollo de las raíces. Sus principales inconvenientes están en el mantenimiento de los emisores para que no se tapone la salida del agua, además de necesitar una fuente constante de agua. Sin embargo, es un sistema muy eficiente que se adapta a numerosos tipos de terreno.

Figura 2.6. Riego por goteo

- Riego hidropónico: es el sistema más eficiente, pero también el más complejo, ya que requiere introducir sustancias químicas y nutritivas en el agua destinada a los cultivos.

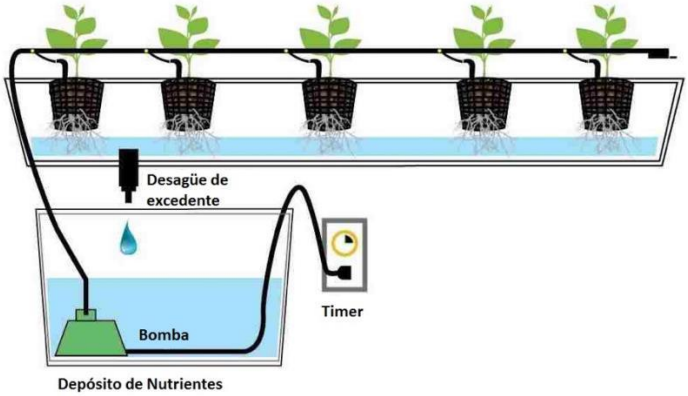


Figura 2.7. Riego hidropónico [11]

- Riego por nebulización: consiste en pulverizar agua para que, en contacto con el aire se evapore y refresque el ambiente. Se utiliza sobre todo en viveros e invernaderos para minimizar el exceso de calor.

Figura 2.8. Riego por nebulización [11]

CAPÍTULO 3. PROBLEMA PLANTEADO Y SOLUCIÓN PROPUESTA

EPÍGRAFE 3.1. PROBLEMA PLANTEADO

Una de las ideas principales de este proyecto, es optimizar al máximo la solución propuesta, es decir, realizar un proyecto cuya aplicación sea sencilla y legible y posea un montaje aplicable a cualquier sistema de riego para pequeñas explotaciones, desde un conjunto de macetas hasta una jardinera de mayor tamaño.

Con esta idea en la cabeza, el problema fue analizar que componentes electrónicos y físicos permitirían llevar a cabo este proyecto.

Para poder desarrollar el proyecto correctamente, hay que definir cuáles son las variables principales que debe de tener un sistema de riego. Estas son la humedad del terreno y la temperatura del ambiente. A su vez debemos de disponer de un controlador que pueda recoger datos de estas variables y proporcionar una respuesta ante cada una de las condiciones. Y por último debemos de disponer de algún elemento que permita la conexión entre nuestro controlador y un dispositivo electrónico.

EPÍGRAFE 3.2. SOLUCIÓN PROPUESTA

Lo primero de todo fue decidir qué componente era el más adecuado para poder realizar la comunicación entre todos los sensores y actuadores de una manera sencilla y permitir llevar un control de todo el sistema de riego. La mejor opción valorando las posibilidades de trabajo, el mercado y los precios fue la placa Arduino UNO R3.

Una vez se ha decidido cual va a ser el controlador de este proyecto, es el momento de elegir cuáles van a ser los sensores y actuadores que permitirán un correcto funcionamiento.

Al pensar en los componentes comentados en el párrafo anterior, también es necesario saber qué protocolo de comunicación se va a emplear y cuál va a ser el canal de comunicación para comprar los componentes con las características acordes a dichos protocolos y canales de comunicación. Después de comparar las opciones de mercado, la mejor forma de controlar el sistema de riego es mediante la ejecución de una aplicación en un dispositivo electrónico que tenga Bluetooth. Por lo que será necesario comprar un módulo Bluetooth y el mejor protocolo de comunicación es I2C. La mejor opción para el módulo Bluetooth elegido es el HC-05.

También son necesarios sensores para medir la humedad del terreno, un sensor que recoja cuál es la temperatura del ambiente y unos actuadores para poder empezar a regar cuando sea preciso. Estudiando las mejores opciones que cumplen las características, los sensores de humedad serán capacitivos con módulo I2C, el sensor de temperatura será una sonda digital de resolución programable y los actuadores serán unas bombas sumergibles con un pequeño motor de corriente continua, activadas mediante relés.

Como se ha elegido la placa Arduino UNO R3 y se necesitan bastantes componentes para el correcto funcionamiento, es necesario añadir una fuente de alimentación externa, ya que la salida de tensión que proporciona la placa no es suficiente para alimentar a todos los componentes.

Para poder hacer una pequeña demostración del funcionamiento se construirá una maqueta que contiene: un módulo Bluetooth HC-05, dos recipientes que simularán una maceta y una jardinera, tres sensores de humedad, una sonda de temperatura, dos relés, dos bombas, la placa de Arduino, un display LCD y una fuente de alimentación externa. Para poder realizar la conexión de todos los componentes también serán necesarias tres placas protoboards.

A continuación se muestra el diagrama de bloques entre los diferentes componentes utilizados. La placa de Arduino es el controlador de toda la maqueta, por el pasan todos los datos utilizados para que este proyecto funcione. Se encarga de recoger los datos obtenidos por los sensores de humedad del medio y la sonda de temperatura, escribirlos en el display LCD para visualizarlos y analizarlos para que, en función de que valores recoja, activar o desactivar los relés que controlan las bombas de riego. A su vez también se encarga de recoger los valores que el dispositivo Bluetooth le envía, para poder ejecutar los diferentes modos de funcionamiento, automático manual o programado.

El diagrama de interconexión e intercambio de datos entre los elementos de la maqueta será el siguiente:

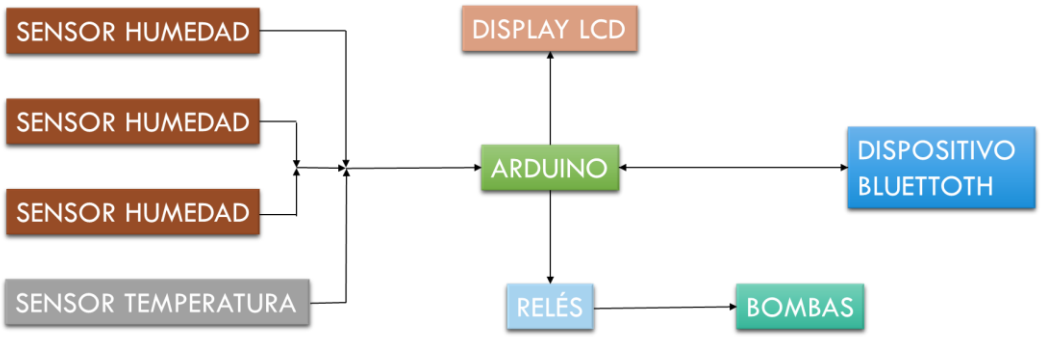


Figura 3.1. Diagrama de interconexión entre elementos

CAPÍTULO 4. IMPLEMENTACIÓN FÍSICA

Para ofrecer una idea visual del comportamiento de este proyecto, se ha llevado a cabo la construcción de una pequeña maqueta donde se reproducirá a pequeña escala el comportamiento de la instalación.

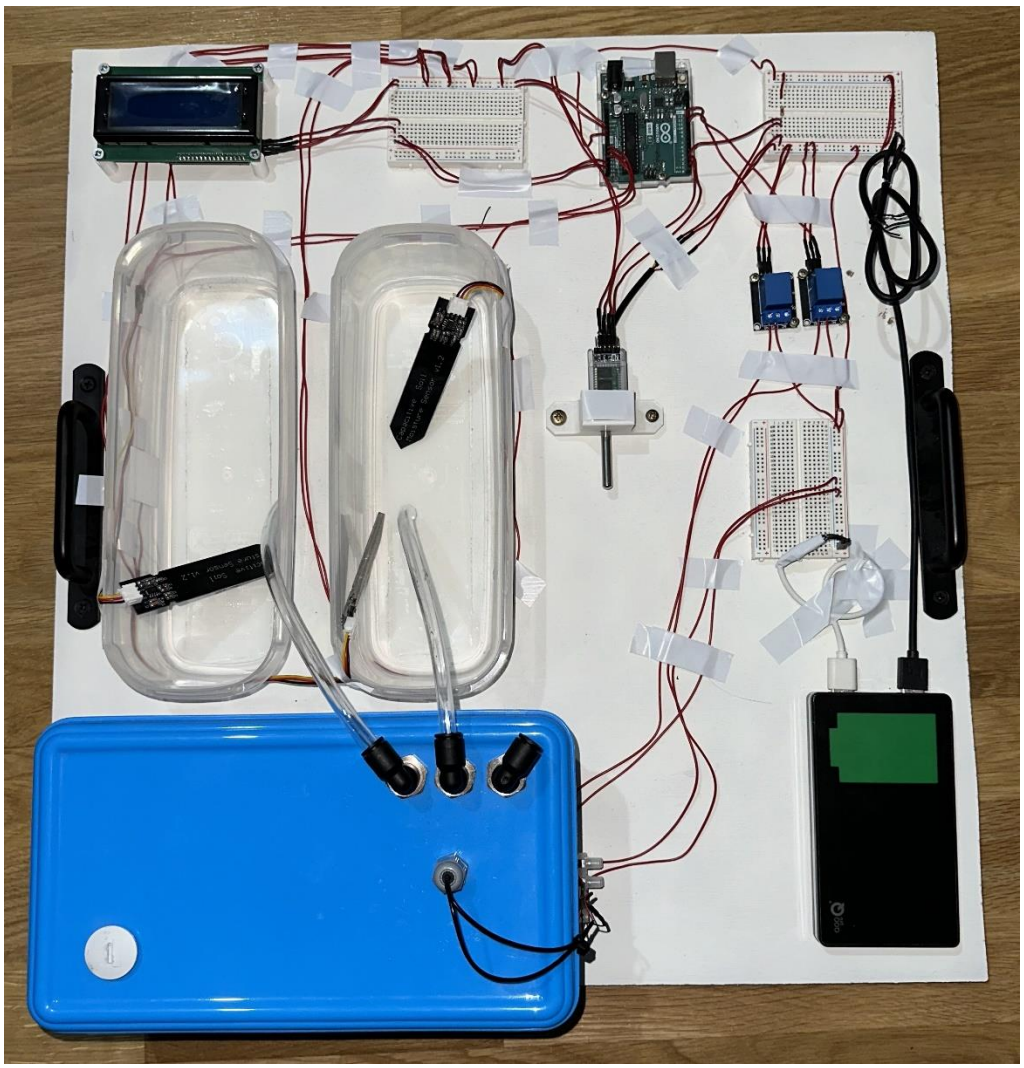


Figura 4.0.1. Maqueta completa – Vista Planta

Realizando una primera descripción de la maqueta podemos ver que está compuesta por tres sensores de humedad, dos recipientes que simulan el cultivo, un depósito donde se almacena el agua que después se usará para regar; en el interior de este hay dos bombas que extraen el agua del depósito y que mediante las pequeñas tuberías de plástico se distribuye a las diferentes zonas de cultivo.

Para poder utilizar estas bombas de riego es necesario la utilización de dos relés que activen o desactiven las bombas en función de la humedad que recogen los sensores. Para poder saber y conocer en todo momento cuál es el

estado del cultivo existe un display LCD que mostrará por pantalla la información básica de la maqueta y un pequeño teclado matricial que permitirá ir navegando por las diferentes pantallas del display.

Todos estos componentes descritos anteriormente se controlan mediante el Arduino UNO R3, que a su vez está gestionado mediante una aplicación móvil. El Arduino es el encargado de extraer la información de los sensores analógicos; a su vez la aplicación móvil es la que se encarga de seleccionar el tipo de riego que queremos en cada momento y mandar al Arduino las señales que tiene que activar o desactivar en función de las condiciones del terreno y del tipo de riego seleccionado. A su vez también podemos navegar por las diferentes pantallas del display LCD.

La alimentación de la maqueta se compone de una pequeña batería portátil que suministra la tensión a las bombas del interior del depósito y a los relés que las pilotan. Por otro lado el Arduino estará conectado a la red eléctrica y será el encargado de suministrar la tensión al display LCD, a los sensores de humedad analógicos y al sensor digital de temperatura.

Cabe destacar que para conectar y distribuir todos los componentes a lo largo de la maqueta ha sido necesaria la utilización de tres placas protoboard.

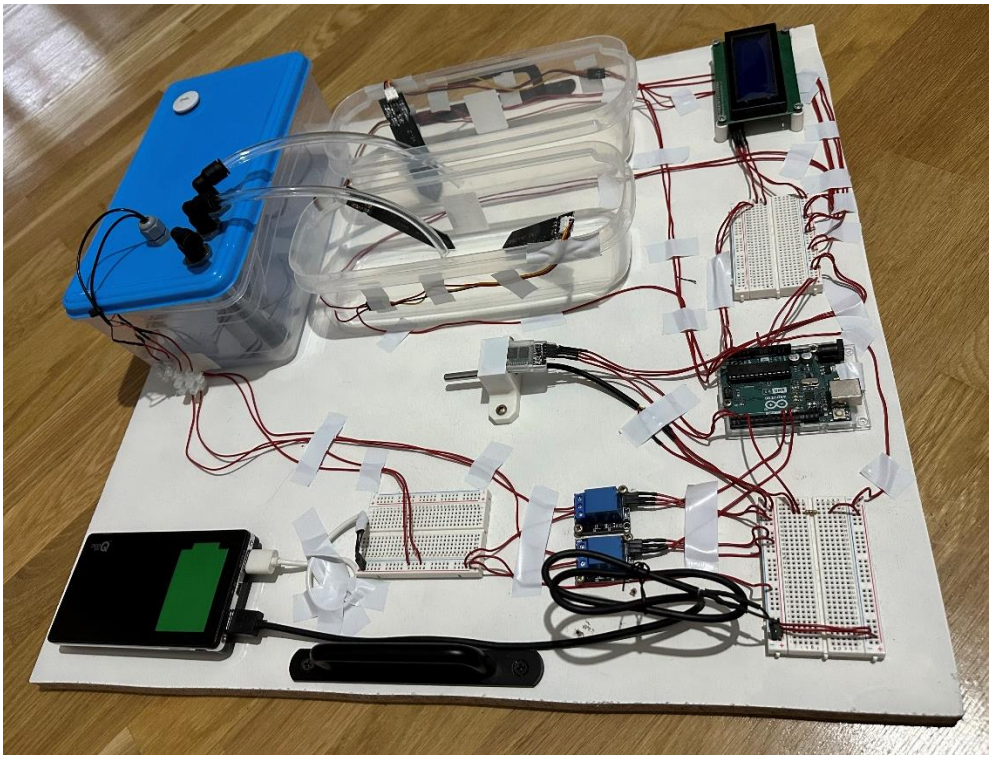


Figura 4.0.2. Maqueta completa – Vista Lateral

En las figuras anteriores se puede ver a grandes rasgos como se distribuyen todos los componentes anteriormente mencionados y sus conexiones. A continuación entramos en más detalle al especificar la información técnica de cada uno de los componentes utilizados.

EPÍGRAFE 4.1. DESCRIPCIÓN DE COMPONENTES

En este epígrafe se explica a modo resumen las características más básicas de cada componente, así como su conexionado.

ARDUINO UNO R3

Arduino UNO R3 es una placa de desarrollo de software basada en el microcontrolador ATmega328P. Es una de las placas más famosas de la marca y su fama se debe a:

- Su precio, ya que es una de las más económicas del mercado.
- Su facilidad de uso, haciendo muy sencilla la implementación de numerosos proyectos.
- Su compatibilidad con cualquier sistema operativo y placas de expansión (shields).
- La disponibilidad de complementos para esta placa es muy alta y se pueden añadir numerosos módulos que le añaden aún más funciones de las que ya posee.

Su durabilidad y seguridad, lo que hace que se pueda utilizar sin miedo de estropearla porque, en el peor de los casos, se podría reemplazar el microcontrolador y seguir utilizándola.

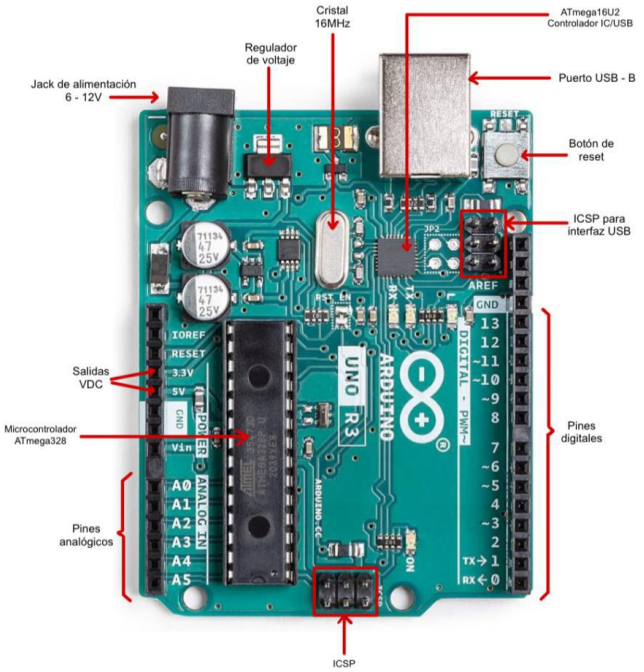


Figura 4.1.1. Arduino UNO R3

En la figura anterior podemos ver una imagen de la placa completa con todas las partes que la componen y una breve descripción que permite conocer un poco más de información sobre su funcionamiento.

La tensión de funcionamiento de la placa si se alimenta por el puerto USB-B es de 5V, sin embargo podemos llegar hasta tensiones de 12V, sin riesgo de dañar la placa, si se alimenta mediante el conectar JACK de alimentación. Esta tarjeta cuenta con tres pines de GND, dos pines de alimentación, uno de ellos de 3,3V y el otro de 5, seis entradas analógicas y catorce pines digitales de entrada y salida. A continuación se insertará una tabla donde se puede ver que nombre se asocia a cada pin y la descripción de que realiza.

PIN / CONECTOR	TIPO	DESCRIPCIÓN
JACK	ALIMENTACIÓN	Alimenta con una tensión de entre 6 y 12V la placa de Arduino
IOREF	ADAPTADOR	Pin que proporciona la señal de referencia a placas de expansión (shields)
3,3V	ALIMENTACIÓN	Permite tener una entrada o salida conectada a 3,3V
5V	ALIMENTACIÓN	Permite tener una entrada o salida conectada a 5V
GND	TIERRA	Pines que proporcionan un valor de tensión y corriente nulo
A0	ANALOG IN	Convierte la señal de un medidor analógico en una señal que varía entre 0 y 5 V
A1	ANALOG IN	Convierte la señal de un medidor analógico en una señal que varía entre 0 y 5 V
A2	ANALOG IN	Convierte la señal de un medidor analógico en una señal que varía entre 0 y 5 V
A3	ANALOG IN	Convierte la señal de un medidor analógico en una señal que varía entre 0 y 5 V
A4/SDA	ANALOG IN/I2C	Convierte la señal de un medidor analógico en una señal que varía entre 0 y 5 V + señal de datos
A5/SCL	ANALOG IN/I2C	Convierte la señal de un medidor analógico en una señal que varía entre 0 y 5 V + señal de reloj
D0/RX	DIGITAL/GPIO	Conexiones entrada/salida digital + comunicación en serie
D1/TX	DIGITAL/GPIO	Conexiones entrada/salida digitals + comunicación en serie
D2	DIGITAL/GPIO	Conexiones entrada/salida digital + entrada para interrupciones
D3	DIGITAL/GPIO	Conexiones entrada/salida digital + PWM + entrada para interrupciones
D4	DIGITAL/GPIO	Conexiones entrada/salida digital
D5	DIGITAL/GPIO	Conexiones entrada/salida digital + PWM
D6	DIGITAL/GPIO	Conexiones entrada/salida digital + PWM
D7	DIGITAL/GPIO	Conexiones entrada/salida digital
D8	DIGITAL/GPIO	Conexiones entrada/salida digital
D9	DIGITAL/GPIO	Conexiones entrada/salida digital + PWM
D10/SS	DIGITAL/GPIO	Conexiones entrada/salida digital + PWM + Slave Select
D11/MOSI	DIGITAL/GPIO	Conexiones entrada/salida digital + PWM + Master Out Salve In
D12/MISO	DIGITAL/GPIO	Conexiones entrada/salida digital + PWM + Master In Slave Out
D13/SCK/LED	DIGITAL/GPIO/LED	Conexiones entrada/salida digital + señal de reloj + led
AREF	DIGITAL	Referencia analógica
SDA	DIGITAL	Señal de datos I2C
SCL	DIGITAL	Señal de reloj I2C
TX	LED	Parpadea cuando se realiza una transferencia de datos mediante el USB
RX	LED	Parpadea cuando se realiza una transferencia de datos mediante el USB
ICSP	ETHERNET SHIELD	Permite la comunicación a través del módulo W5100 utilizando el bus de datos SPI
RESET	RESET	Permite el reinicio del controlador (el puntero se coloca en el inicio del código)
PUERTO USB-B	CONEXIÓN/ALIMENTACIÓN	Permite la transferencia de archivos entre un PC y la placa + alimentación de la placa

Figura 4.1.2. Tabla de descripción de pines de la tarjeta Arduino

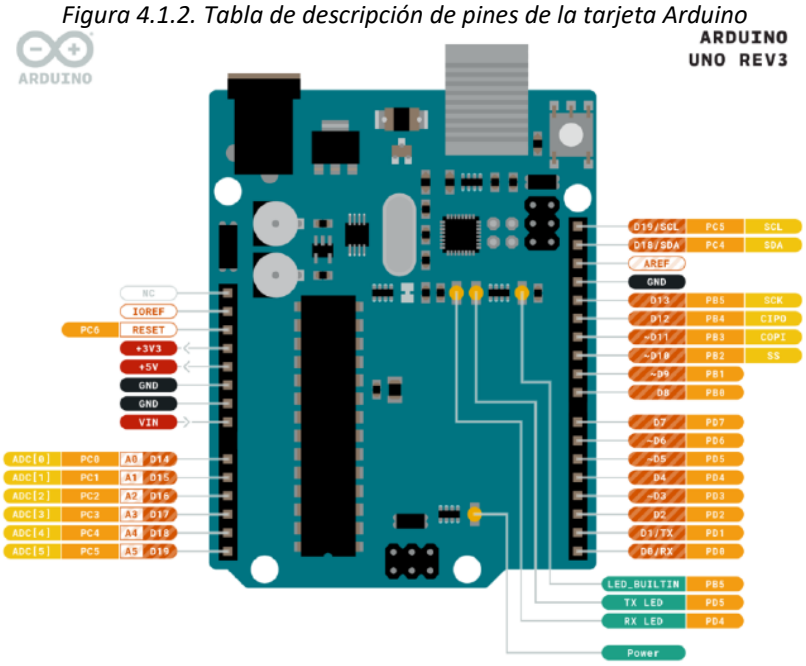


Figura 4.1.3. Ubicación de pines [15]

A modo de profundizar un poco más en la descripción de esta placa de Arduino, podemos decir que está basada en el microcontrolador ATMega328P. Es un controlador de alto rendimiento de 8 bits, basado en la tecnología AVR RISC.

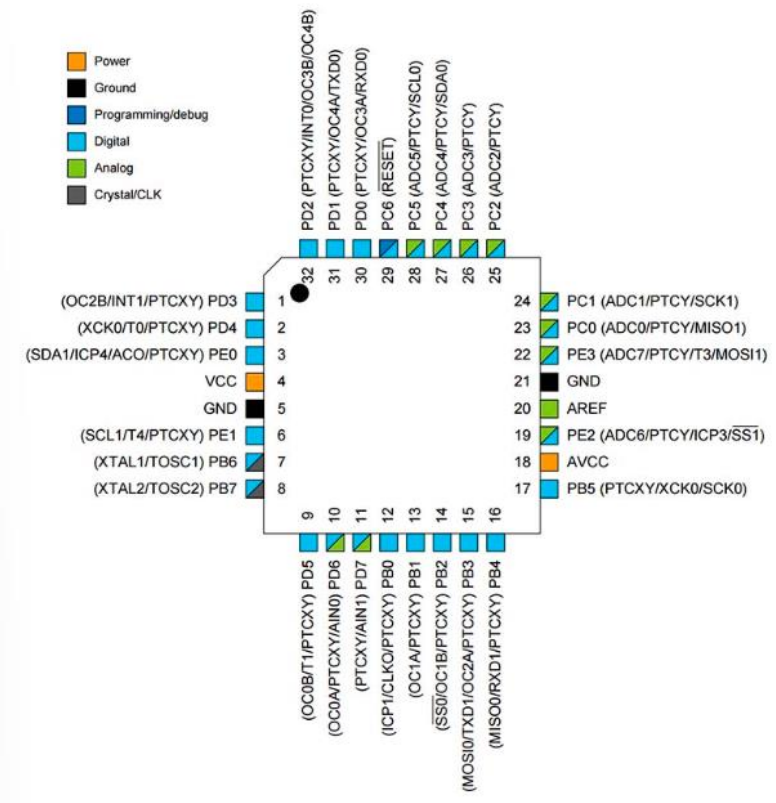


Figura 4.1.4. Encapsulado TQFP del microcontrolador ATMega328P [13]

Tiene 23 pines de entradas y salidas, un convertidos A/D con 6 canales, una USART programable, una interfaz I2C y un puerto SPI. Dispone de 32 kB de memoria flash que se destinan a almacenar el programa (sketch), 2 kB de memoria SRAM donde se almacenan las variables declaradas y 1 kB de memoria EEPROM que almacena datos ante posibles fallos de alimentación. Además, dispone de un cristal oscilador y de un resonador cerámico de 16MHz, utilizados por el ATmega16Us y el ATmega328P, respectivamente, para generar las señales de reloj. Su funcionamiento se basa en analizar las variaciones de voltaje para generar una onda cuadrada de la misma frecuencia. Esta onda determina la velocidad a la que el microcontrolador ejecuta las acciones programadas en él.

Este Arduino ha experimentado grandes y numerosas actualizaciones con respecto a sus antepasados, actualmente es la placa más robusta y popular de Arduino.

Podemos diferenciar las evoluciones de las placas de Arduino en dos etapas claramente diferenciadas, la primera se componía de Arduino Seria, en la cual la comunicación con el dispositivo programador se realizaba mediante el protocolo de comunicación en serie asíncrono RS232; y la última que es Arduino USB, en la que se sustituye el protocolo RS232 por un puerto USB para simplificar la conexión con el dispositivo programador.

Como se necesita disponer de una comunicación serie entre el Arduino y el dispositivo programador, el puerto USB utiliza los pines RX y TX (pines digitales) para poder recibir y transmitir los datos. [13]

SENSOR DE TEMPERATURA

El termómetro digital es un sensor que mide temperaturas en el rango de -55ºC a 125ºC. Gracias a que es un sensor digital la señal es mucho más fiable. Utiliza el protocolo de comunicación OneWire para poder comunicarse con la placa Arduino. Es necesario colocar una resistencia de 4,7 kΩ entre la alimentación y la señal digital que se conecta con el pin 2 del Arduino para poder realizar una medición correcta. Es un dispositivo que solo tiene tres pines, alimentación, tierra y señal.

En la siguiente figura se muestra cómo se realizan las conexiones entre la placa de Arduino y el sensor de temperatura.

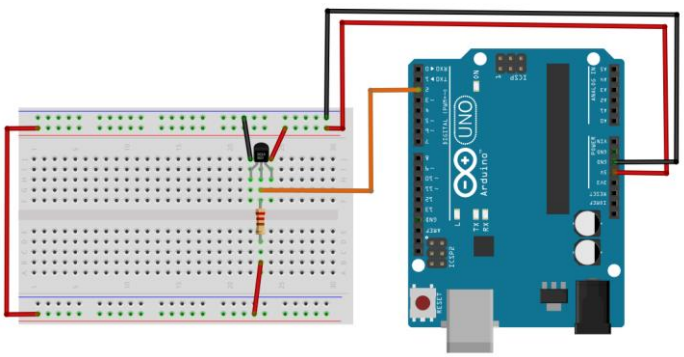


Figura 4.1.5. Esquema eléctrico del sensor de temperatura digital

SENSOR DE HUMEDAD

El sensor capacitivo de humedad de suelo es el que va a recoger los datos de la humedad del terreno mediante señales analógicas que se convierten al rango de tensión de 5V. Es un dispositivo que solo tiene tres pines, alimentación, tierra y señal analógica y está diseñado específicamente para trabajar entre una tensión de alimentación entre 3,3 y 5V. Su corriente de operación es de 5 mA por lo que es ideal ya que el Arduino proporciona una corriente muy pequeña.

En la siguiente página se muestra cómo se realizan las conexiones entre la placa de Arduino y los cuatro diferentes sensores de humedad.

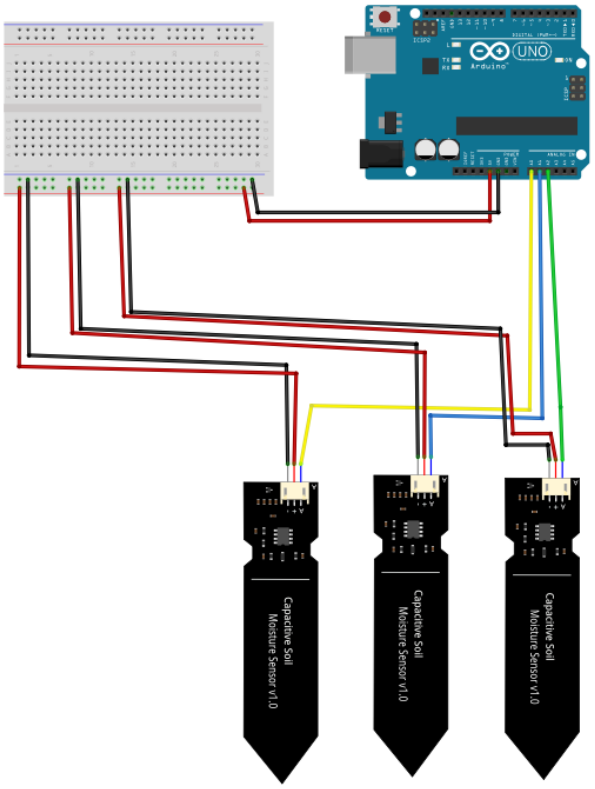


Figura 4.1.6. Esquema eléctrico de los sensores de humedad capacitivos

BOMBAS EXTRACTORAS

Las bombas colocadas en el interior del depósito son las encargadas de extraer el agua de este; gracias a unas pequeñas tuberías de plástico son capaces de distribuirla hasta sus respectivas macetas. Funcionan en un rango de tensión de 3 a 5V, pudiendo regularse su potencia mediante un potenciómetro y conseguir variar el flujo de riego. La maqueta dispone de tres bombas colocadas en el interior del depósito de agua que son alimentadas por una fuente de alimentación externa, en este caso, una batería portátil de 5V con dos salidas, una proporciona 1A mientras que la otra proporciona 2.1A. En nuestro caso da igual a que puerto de carga conectarlas, ya que el consumo de corriente de estas bombas es del orden de miliamperios.

RELÉS

Son los encargados de activar o cortar el suministro de tensión a las bombas, por lo que si el relé se activa, la bomba que está controlada por ese relé comenzará a funcionar y a regar la maceta correspondiente; cuando el relé corte la tensión, la bomba se apagará y dejará de regar. En la maqueta disponemos de tres relés conectados a una alimentación externa (batería portátil), a la placa de Arduino para poder escribir en las señales digitales y a las bombas que controlan. Al igual que en el caso de alimentación de las bombas da igual a que puerto de carga conectar estos relés, ya que el consumo de corriente también es del orden de miliamperios.

A continuación, podemos ver una captura del esquema eléctrico tanto de las bombas como de los relés utilizados en esta maqueta, ya que se tienen que conectar juntas para que los relés controlen la tensión de las bombas.

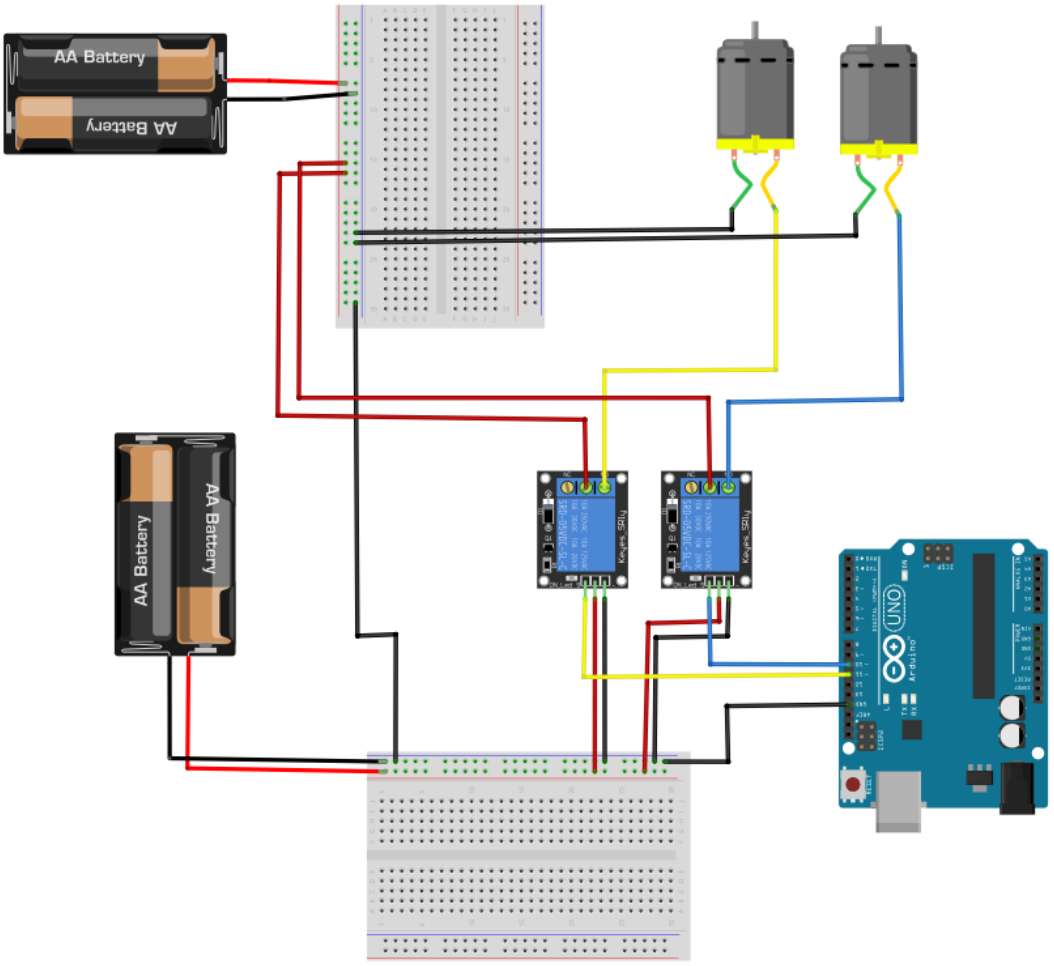


Figura 4.1.7. Esquema eléctrico de las bombas y relés

BATERÍA – ALIMENTACIÓN AUXILIAR

Para poder alimentar a las bombas de tensión, sin que la instalación llegue a sus límites de funcionamiento se ha utilizado una batería portátil de 10000mA de capacidad que suministra una tensión de 5V a 2.1A a las bombas extractoras y 5V y 1A a los relés que permiten que las

bombas se enciendan o se apaguen. En la Figura 3.1.5 se representa esta batería auxiliar separada en dos pilas de baterías.

DISPLAY LCD I2C

Para poder mostrar por pantalla la información de la forma más detallada posible, el display elegido para esta maqueta es un display LCD de 20x4 con comunicación I2C. Con este módulo solo es necesario conectar cuatro cables a la placa de Arduino para que se realice la comunicación, lo que nos facilita la conexión. Sin embargo es necesario conectar el módulo de comunicación I2C al display LCD mediante las conexiones que se ven en la siguiente foto.

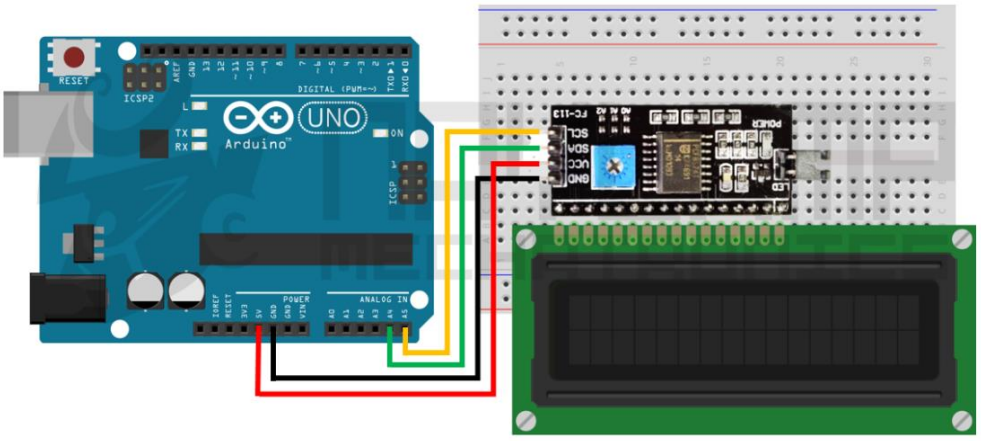


Figura 4.1.8. Esquema eléctrico del display LCD con comunicación I2C

EPÍGRAFE 4.2. PROTOCOLO DE COMUNICACIÓN I2C

I2C es un protocolo de comunicación serial que define las tramas de datos y las conexiones físicas para transferir datos entre dos o más dispositivos digitales. Se trata de un bus maestro - esclavo que requiere únicamente dos conexiones, la señal de reloj (CLK/SCL) y la señal de datos (SDA).

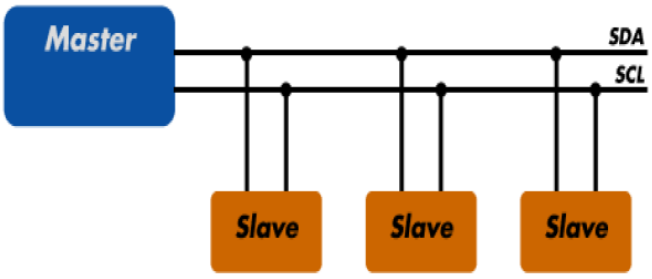


Figura 4.2.1. Bus de campo I2C

El protocolo I2C es uno de los más utilizados para comunicarse con sensores digitales, ya que a diferencia del puerto Serial, su arquitectura permite tener una confirmación de los datos recibidos (ACK), además permite mandar mensajes más completos y detallados ya que sus tramas (mensajes) pueden tener una mayor longitud de bits.

Cabe destacar que este protocolo tiene un bus de comunicación síncrono, es decir que todos el maestro y los esclavos tienen la misma velocidad de transmisión y recepción de datos.

Cada dispositivo tiene una dirección única dentro de este bus, por lo que se puede acceder a cada uno de ellos de forma individual.

El maestro se encarga de iniciar y finalizar la comunicación, una vez que esta se ha iniciado, envía una trama de 7 bits de dirección, el octavo bit que es el de menor peso, indica si el maestro quiere enviar o recibir información (si el último bit es un 0 quiere decir que será escritura, mientras que si es un 1 será una operación de lectura). Una vez que el esclavo recibe esta cadena de 8 bits compara la dirección con la suya propia, si es la misma envía un bit del tipo ACK (bit de confirmación de recepción de datos), mientras que si no lo es lo ignora y sigue pendiente de recibir nuevas cadenas de bits para comparar con su dirección.

Una vez el maestro recibe el bit de confirmación la comunicación esta lista para producirse y dependiendo del último bit de la primera cadena de bits el esclavo enviará información al maestro y este le devolverá el ACK o será el maestro el que envíe información al esclavo y sea este quien deba devolver el ACK. Una vez que el maestro da por finalizada la comunicación, envía una condición de stop a los esclavos y así el bus queda libre. [19]

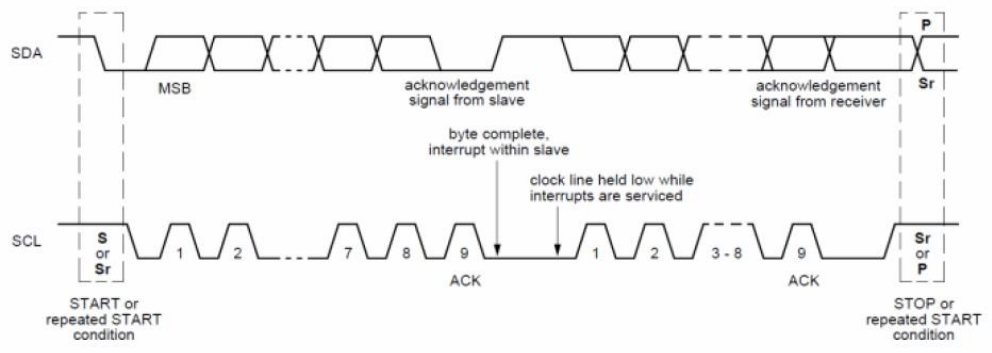


Figura 4.2.2. Transmisión de datos I2C

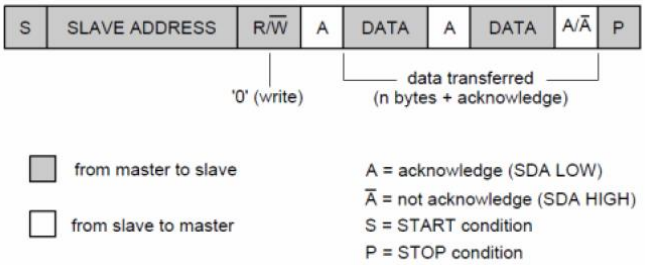


Figura 4.2.3. Tipo de direcciones de transferencia dentro del bus I2C [19]

EPÍGRAFE 4.3. PRESUPUESTOS MAQUETA E INSTALACIÓN REAL

A continuación queda reflejado en una tabla el coste total que ha supuesto construir esta pequeña maqueta. En los Anexos estarán disponibles las hojas de características de todos los componentes empleados.

Los costes por unidad no son exactos, ya que cuanta más cantidad de un artículo compres, el precio disminuye. En este presupuesto, a su vez de los materiales empleados, quedan reflejadas las horas que se han empleado en llevarla a cabo, y por tanto su coste asociado.

PRESUPUESTO MAQUETA			
ARTÍCULO	**Nº UNIDADES/Horas**	**€/Ud/h**	**TOTAL/ARTÍCULO**
Arduino Uno R3	1	29,28	29,28
Modulo Bluetooth BT-05	1	9,99	9,99
Micro Bombas Sumergibles	3	3	9
Display LCD 20*4 I2C	1	12,99	12,99
Pines Macho-Hembra	1	8,48	8,48
Protoboard 400 pines	3	2,95	8,85
Sensores Humedad Capacitivos	5	1,8	9
Relés KY-019 5V	3	2,4	7,2
Racores	4	4,5	18
Batería Alimentación Externa	1	40,98	40,98
Envases / Recipientes	4	3,25	13
Cables Tensión	2	3,5	7
Base Maqueta	1	12	12
Asas	2	6,98	13,96
Horas Destinadas	200 h	20	4000

TOTAL	**4199,73**

A diferencia del presupuesto anterior, a continuación se representa cuál sería el presupuesto real con unos componentes de mejor calidad, más potentes y versátiles.

Al igual que para el presupuesto de la maqueta, los costes por unidad no son exactos, ya que cuanta más cantidad de un artículo compres, el precio disminuye.

Al tener componentes con mejores prestaciones el precio de estos es más elevado, sin embargo, con el estudio previo hecho anteriormente para diseñar y construir esta maqueta reduciría los costes de diseño y montaje porque ya existe una base sólida sobre la que partir de futuros diseños

PRESUPUESTO			
Instalación Real			
ARTÍCULO	**Nº UNIDADES**	**€/Ud**	**TOTAL/ARTÍCULO**
Arduino Mega	1	42	42
Modulo Bluetooth BT-05	1	9,99	9,99
Bomba agua 12Vdc 3.6W 2 Pines Waterproof	4	11,46	45,84
Display LCD 40*4	2	75,43	150,86
Modulo I2C	5	5	25
Cables Tensión	3	8,48	25,44
Protoboard 400 pines	5	2,95	14.75
Sensores Humedad Capacitivos	10	5	50
Relés KY-019 12V	3	4,80	14,4
Racores	20	4,5	90
Batería Alimentación	5	40,98	204,9
Estudio y Montaje	10 h	30	3000
			0

TOTAL	**3673,18**

CAPÍTULO 5. ARDUINO IDE

Para poder realizar este proyecto, he elegido la herramienta Arduino IDE porque pienso que en el momento actual es la mejor para poder controlar y programar un Arduino.

EPÍGRAFE 5.1. DIAGRAMA DE FLUJO

A continuación se adjunta el diagrama de flujo que representa el funcionamiento de esta pequeña instalación de riego.

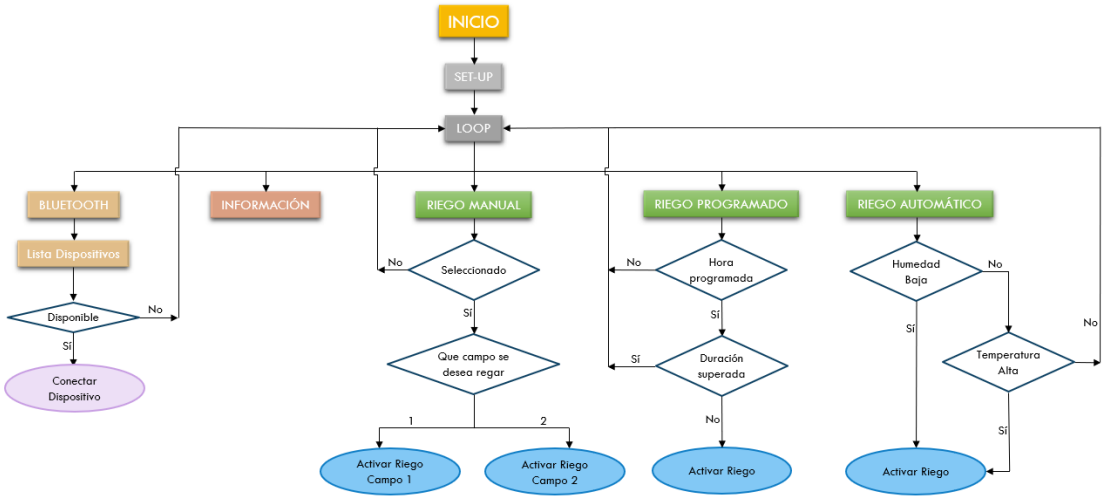


Figura 5.1.1. Diagrama de flujo

A modo de resumen, aunque más adelante se explicará en detalle, el proyecto consta de cuatro ramas principales, que son los tres tipos de riego disponibles, manual, automático sensorial o automático programado; la conexión mediante Bluetooth; y una rama secundaria, desde la cual se elegirá poder ver la información de un campo u otro en el display LCD. El riego manual se activa si el usuario selecciona riego manual y elige regar un campo en concreto o varios, el riego automático sensorial se activa dependiendo de unas determinadas condiciones atmosféricas y el riego automático programado si el usuario lo selecciona se activa a la hora programada y durante el tiempo establecido.

EPÍGRAFE 5.2. PROGRAMA

A continuación, se muestra y explica el programa desarrollado en Arduino IDE en profundidad.

BIBLIOTECAS

```
//---------------------------------DEFINICIÓN DE FUNCIONES-------------------------------------//
//// BIBLIOTECAS ////
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <OneWire.h>
#include <DallasTemperature.h>
//////////////////////
```

Figura 5.2.1. Programa Arduino – Bibliotecas

En este programa se incluyen dos bibliotecas básicas, una es la biblioteca “Wire.h” que se utiliza para poder enviar y recibir información con buses de campo I2C; y la biblioteca “LiquidCrystal_I2C.h” que es necesaria para poder mostrar textos e información en la pantalla de display LCD incorporado en esta maqueta.

CONSTANTES Y VARIABLES

```
/////// DEFINIR DISPLAY Y CONSTANTES ////////
LiquidCrystal_I2C lcd = LiquidCrystal_I2C (0x27,20,4);
byte gcent[9] = { B01110, B01010, B01110, B00000, B00000, B00000, B00000, B00000 };

// Data wire is plugged into port 2 on the Arduino
#define ONE_WIRE_BUS 2

// Setup a oneWire instance to communicate with any OneWire devices (not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

float temperatura;
int opc;
double tiempo_riego=0;
double porcentaje_medio;
char letra;
bool man;
bool aut;
bool prog;
bool primer_riego_prog;
bool fin_riego_prog;
bool campo1;
bool campo2;

//// DATOS, CONSTANTES Y VARIABLES SENSORES ////
  // SENSOR HUMEDAD 1 //
  const int valor_seco1 = 906;
  const int valor_humedo1 = 704;
  float lectura_sensor1;
  double porcentaje1;
  // SENSOR HUMEDAD 1 //
  const int valor_seco2 = 905;
  const int valor_humedo2 = 724;
  float lectura_sensor2;
  double porcentaje2;
  // SENSOR HUMEDAD 1 //
  const int valor_seco3 = 901;
  const int valor_humedo3 = 726;
```

```
  float lectura_sensor3;
  double porcentaje3;
  // SENSOR HUMEDAD 1 //
  const int valor_seco4 = 903;
  const int valor_humedo4 = 708;
  float lectura_sensor4;
  //double porcentaje4;

//// DATOS, CONSTANTES Y VARIABLES DE BOMBAS ////
  int bomba1 = 11;
  int bomba2 = 10;
  //int bomba3 = 12;
  int termometro = 2;
////////////////////////////////////////////////
```

Figura 5.2.2. Programa Arduino – Declaración de constantes y variables

Se definen cuáles son las variables que determinan el rango de medición de cada sensor de humedad, las variables globales que determinan el modo de funcionamiento del riego y se definen los pines en los que están conectados las bombas para una mayor accesibilidad en el caso de que se produzca un cambio en la conexión.

PROGRAMA

```
//// RIEGO AUTOMÁTICO ////
void riego_aut (int bomba, double porcentaje,float temperatura) {

  lcd.setCursor(1, 3);
  lcd.print("RIEGO: ");
  lcd.setCursor(8, 3);
  lcd.print("AUTOMATICO  ");

  digitalWrite(bomba,LOW);

  if (porcentaje < 30)
  {
    digitalWrite(bomba,HIGH);
  }
  else if(porcentaje > 50)
  {
    digitalWrite(bomba,LOW);
  }
  else if(porcentaje > 50 && temperatura > 26)
  {
    digitalWrite(bomba,HIGH);
  }

}
//////////////////////////
```

Figura 5.2.3. Programa Arduino – Función de riego automático

La captura de imagen adjuntada anteriormente corresponde a la función que permite activar el riego automático sensorial. Es decir, esta función evalúa cuál es el porcentaje de humedad y temperatura del terreno y en función de estas, activa o desactiva automáticamente el riego automático.

A continuación se muestran las diferentes funciones que puede desarrollar el programa en función de la opción que sea seleccionada por el usuario.

```
//// MODOS DE FUNCIONAMIENTO ////
void modo (char numero) {

  lcd.setCursor(0, 0);
  lcd.print("SELECCIONA UN CAMPO ");
  lcd.setCursor(0, 1);
  lcd.print("TEMPERATURA: ");
  lcd.setCursor(19, 1);
  lcd.write(byte(0));
  lcd.setCursor(14, 1);
  lcd.print(temperatura);

  lcd.setCursor(4, 2);
  lcd.print("HUMEDAD: ");
  lcd.setCursor(19, 2);
  lcd.print("%");

  lcd.setCursor(1, 3);
  lcd.print("RIEGO: ");

  //// VISULALIZACIÓN CAMPO 1 ////
  if (numero == 's') {
    lcd.setCursor(0, 0);
    lcd.print("         ");
    lcd.setCursor(7, 0);
    lcd.print("CAMPO 1     ");
    lcd.setCursor(14, 2);
    lcd.print(porcentaje1);
    campo1 = true;
    campo2= false;
  }
  //// VISULALIZACIÓN CAMPO 2 ////
  else if (numero == 'd') {
    lcd.setCursor(0, 0);
    lcd.print("         ");
    lcd.setCursor(7, 0);
    lcd.print("CAMPO 2     ");
    lcd.setCursor(14, 2);
    lcd.print(porcentaje_medio);
    campo1 = false;
    campo2= true;
  }
  ////////////////////////////////
```

Figura 5.2.4. Programa Arduino – Opciones de usuario 1

Se puede ver la información de los dos campos en el display LCD incorporado a la maqueta.

```
  //// RIEGO AUTOMÁTICO SENSORIAL ////
  else if (letra=='a' || aut == true){
    aut = true;
    prog = false;
    man = false;
    riego_aut(bomba1,porcentaje1,temperatura);
    riego_aut(bomba2,porcentaje_medio,temperatura);
  }
  ////////////////////////////////////

//// RIEGO MANUAL ////
  else if(letra=='m'||man==true){
    prog = false;
    man = true;

    if (aut == true){
      aut = false;
      digitalWrite(bomba1,LOW);
      digitalWrite(bomba2,LOW);
    }
```

```
  else if(letra=='w'){
    man = false;
  }

  lcd.setCursor(1, 3);
  lcd.print("RIEGO: ");
  lcd.setCursor(8, 3);
  lcd.print("MANUAL      ");

  //if(letra=='m'&& aut==false && prog==false){
    //digitalWrite(bomba1,LOW);
    //digitalWrite(bomba2,LOW);
  //}
  if(letra=='k'){
    digitalWrite(bomba1,HIGH);
  }
  else if(letra=='o'){
    digitalWrite(bomba1,LOW);
  }
  else if(letra=='j'){
    digitalWrite(bomba2,HIGH);
  }
  else if(letra=='i'){
    digitalWrite(bomba2,LOW);

  }
}
//////////////////////

 //// RIEGO AUTOMÁTICO PROGRAMADO ////
else if (letra=='w'||prog == true){

  aut = false;
  man = false;
  prog = true;

  lcd.setCursor(1, 3);
  lcd.print("RIEGO: ");
  lcd.setCursor(8, 3);
  lcd.print("PROGRAMADO  ");

  //if (prog == true){

    digitalWrite(bomba1,LOW);
    digitalWrite(bomba2,LOW);


    if (aut == true && man == true){
     aut = false;
      man = false;
      digitalWrite(bomba1,LOW);
      digitalWrite(bomba2,LOW);
    }

    else if (letra=='2'){
     tiempo_riego = 5000;
    }
    else if (letra=='3'){
      tiempo_riego = 10000;
    }

    else if (letra=='4'){
      tiempo_riego = 20000;
    }
    else if (letra=='5'){
     tiempo_riego = 30000;
    }
    else if (letra=='6'){
    tiempo_riego = 40000;
    }
    if (letra=='v') {
      digitalWrite(bomba1,HIGH);
      digitalWrite(bomba2,HIGH);
```

```
        Serial.println(tiempo_riego);
        delay(tiempo_riego);

        digitalWrite(bomba1,LOW);
        digitalWrite(bomba2,LOW);

        delay(60000-tiempo_riego);
      }
    //}
  }
  ////////////////////////////////////////
```

Figura 5.2.5. Programa Arduino – Opciones de usuario 2

En las figuras adjuntas catalogadas como opciones de usuario 1 y 2 podemos ver como el programa es capaz de detectar y activar cuál es el tipo de riego que el usuario desea realizar en cualquier momento.

La instalación se enciende y arranca sin tener seleccionado ningún modo.

- Si el usuario selecciona el tipo de riego automático sensorial, esté tipo quedará activo hasta que el usuario seleccione otro tipo de riego. A su vez este modo solo activa el riego cuando la humedad del terreno este por debajo del 30% y lo desactivará una vez que este haya alcanzado un nivel mínimo del 50%, siempre y cuando la temperatura sea menor de 26ºC, sino se continuará regando hasta alcanzar una humedad del 70%.
- Si el usuario activa el tipo de riego automático programado, deberá definir la hora a la que quiera que el riego se active y el tipo que deba de estar activo.
- Si el usuario activa la petición de modo manual, este tipo de riego será activado y el Arduino esperará a que el operario pulsa cuál/es de los campos desea regar.

CAPÍTULO 6. Aplicación Móvil

Para poder realizar este proyecto, he elegido la herramienta MIT APP INVENTOR (herramienta de desarrollo de software online, vía navegador web) porque pienso que es fácil y sencillo de trabajar con ella; a su vez, también la he elegido ya que no hemos estudiado ni manejado esta herramienta durante la estancia en la universidad, por lo que veía una vía para poder adquirir nuevos conocimientos y desarrollos.

El programa dispone de cinco menús principales, en los que se detallará, con más profundidad, su funcionamiento más adelante. Y es capaz de controlar tres tipos diferentes de riego.

La comunicación que se realiza entre el dispositivo que posee la aplicación web, desde donde se controla el riego, y el dispositivo Bluetooth conectado a la placa de Arduino, se hace mediante la comunicación serial definida en los puertos digitales 0 y 1 de la placa del Arduino.

EPÍGRAFE 6.1. PANTALLA DE INICIO

A modo de introducción y acceso a la aplicación, se ha creado una pantalla de inicio, en la que se refleja el autor y las instituciones implicadas en ella.

Figura 6.1.1. Aplicación Móvil – Pantalla Inicio

EPÍGRAFE 6.2. PANTALLA PRINCIPAL

En la siguiente captura de pantalla, se muestra la apariencia de la pantalla principal al iniciar la aplicación. Se pueden diferenciar cinco menús, de los que se hablarán a continuación con un poco más de detalle.

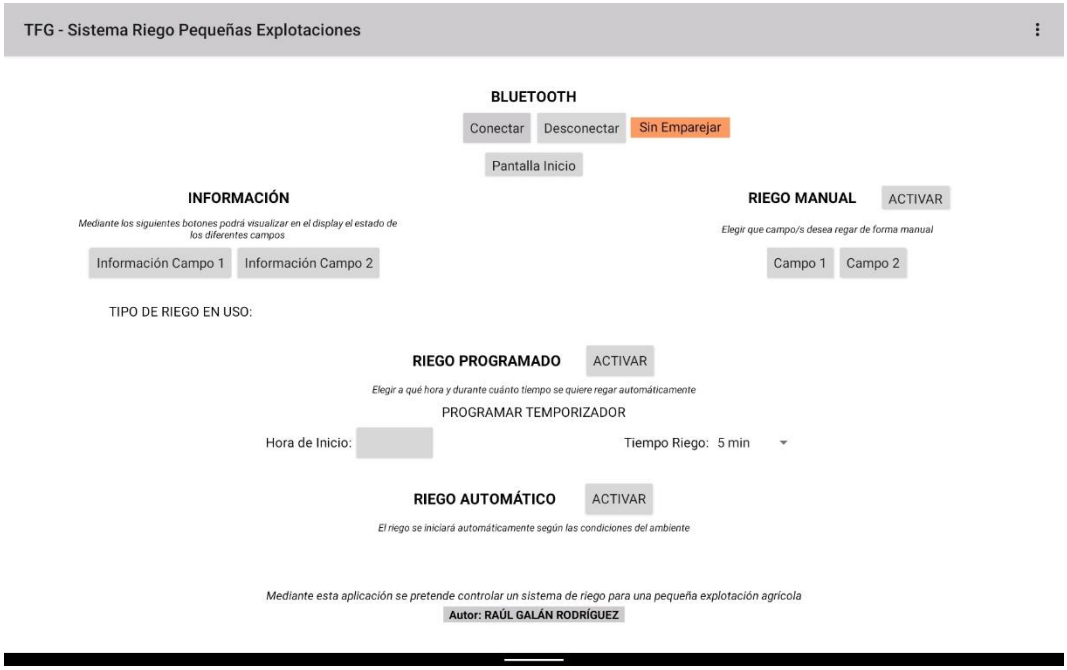


Figura 6.1.2. Aplicación Móvil – Pantalla Principal

BLUETOOTH

Este primer menú situado en la parte central superior de la captura de pantalla anterior sirve para poder realizar la conexión o desconexión con el dispositivo Bluetooth conectado al Arduino, ver el estado de esta conexión y viajar hasta la pantalla de inicio donde regresaríamos de nuevo a la *Figura 5.1.2.*

INFORMACIÓN

Desde este menú es posible cambiar la información que queremos que aparezca en el display LCD conectado al Arduino, a la vez que nos indica que tipo de riego está en uso en cada momento.

RIEGO MANUAL

Este riego permite seleccionar un riego manual a ambos campos. Se pueden regar por separado o juntos durante el tiempo que se desee. Su activación o desactivación siempre se realizan de forma manual y es necesario estar presente en la instalación para poder realizarlo.

RIEGO PROGRAMADO

Mediante este menú, podemos definir la hora a la que queremos que se comience a regar y la duración que queremos que tenga el riego. Para que este riego se active, es necesario que se seleccione una hora de comienzo y la duración; sino el riego programado estaría habilitado pero nunca se pondría en marcha. A diferencia del riego manual, si se selecciona este tipo de riego no es necesario estar presente en la instalación a la hora de inicio ni de finalización del riego.

RIEGO AUTOMÁTICO

Si se selecciona este tipo de riego, la placa Arduino leyendo y analizando las variables de temperatura y humedad que recibe de los sensores, decidirá cuando se debe de activar el riego o detenerlo. A diferencia del riego manual, y al igual que el riego programado, si se selecciona este tipo de riego no es necesario estar presente en la instalación a la hora de inicio ni de finalización del riego.

EPÍGRAFE 6.3. BLOQUES DE CÓDIGO

A continuación en las siguientes capturas de pantalla se podrá ver cuál es la lógica que sigue cada uno de los elementos que forman la aplicación. No se entrará muy en detalle porque es muy visual.

BLUETOOTH

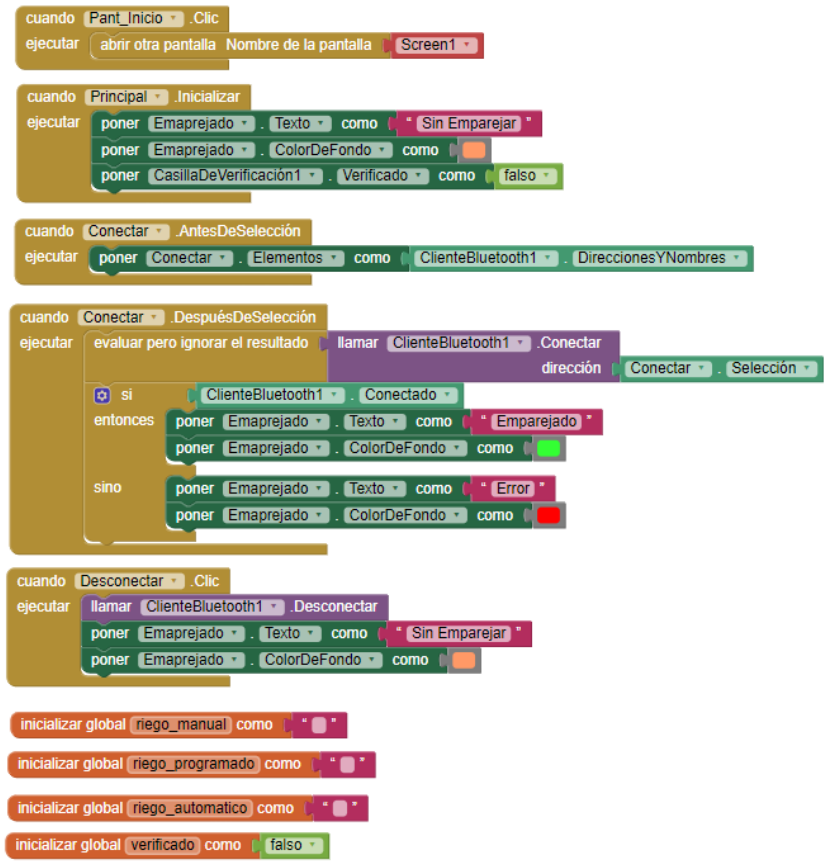


Figura 6.3.1. Aplicación Móvil – Bluetooth

En esta primera captura, se inicializa la pantalla de la aplicación, se realiza la conexión Bluetooth y se declaran unas variables que serán importantes para el flujo del programa.

INFORMACIÓN

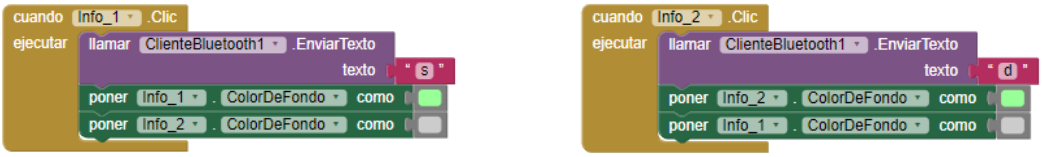


Figura 6.3.2. Aplicación Móvil – Información

Mediante estos botones, se puede en el display la información del campo que queramos, dependiendo de cual pulsemos.

RIEGO MANUAL

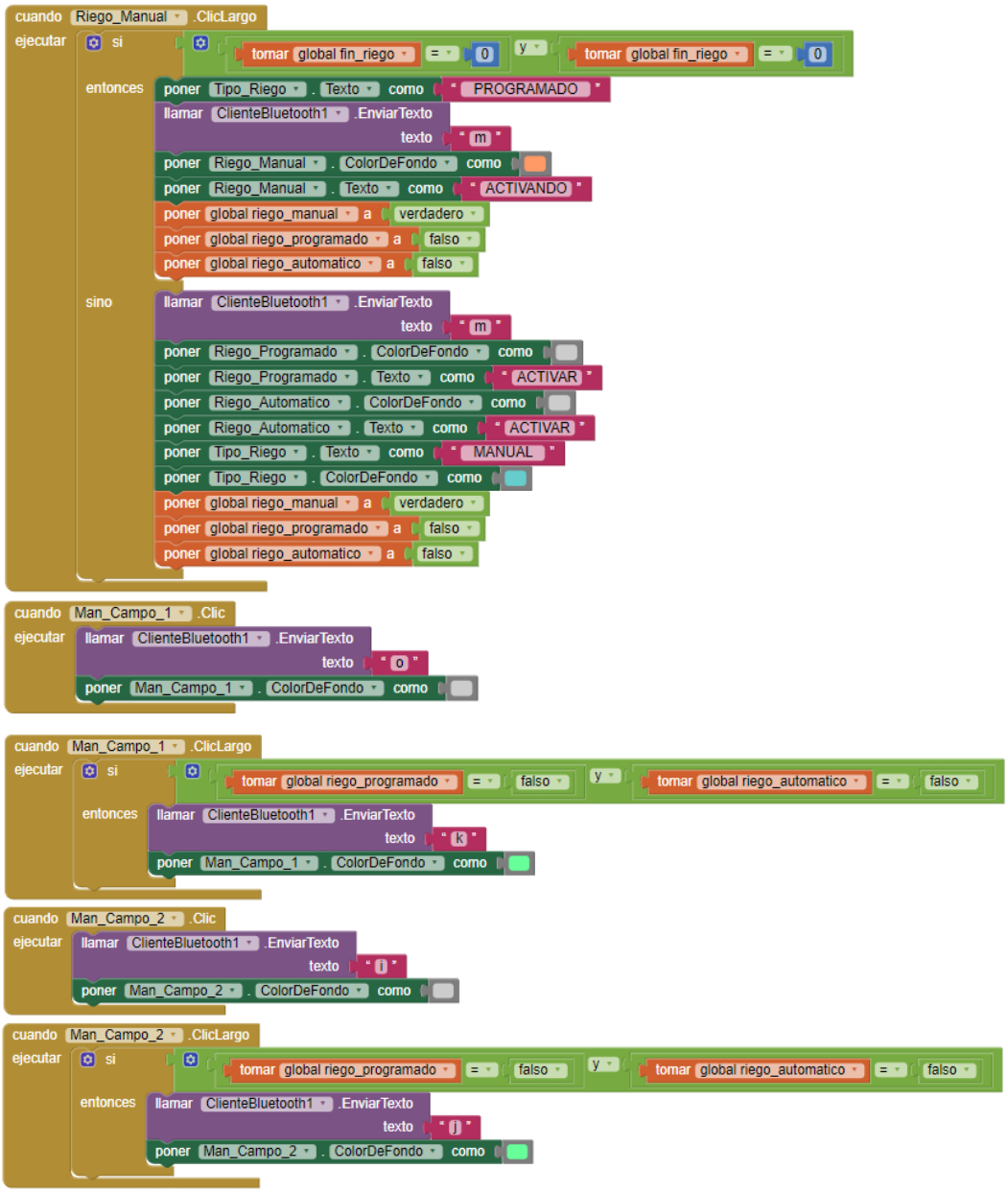


Figura 6.3.3. Aplicación Móvil – Riego Manual

El riego manual queda seleccionado cuando se mantiene pulsado el botón de activación. Para poder regar los campos es necesario seleccionar cual de ellos queremos regar. Se activa el riego en cada uno de los campos manteniendo pulsado el botón por unos segundos y se desactiva solo con un toque. Las demás acciones que se ven son propias de la representación de condiciones en pantalla para una mejor visualización y activación de señales que se envían al Arduino para el control del sistema.

RIEGO PROGRAMADO

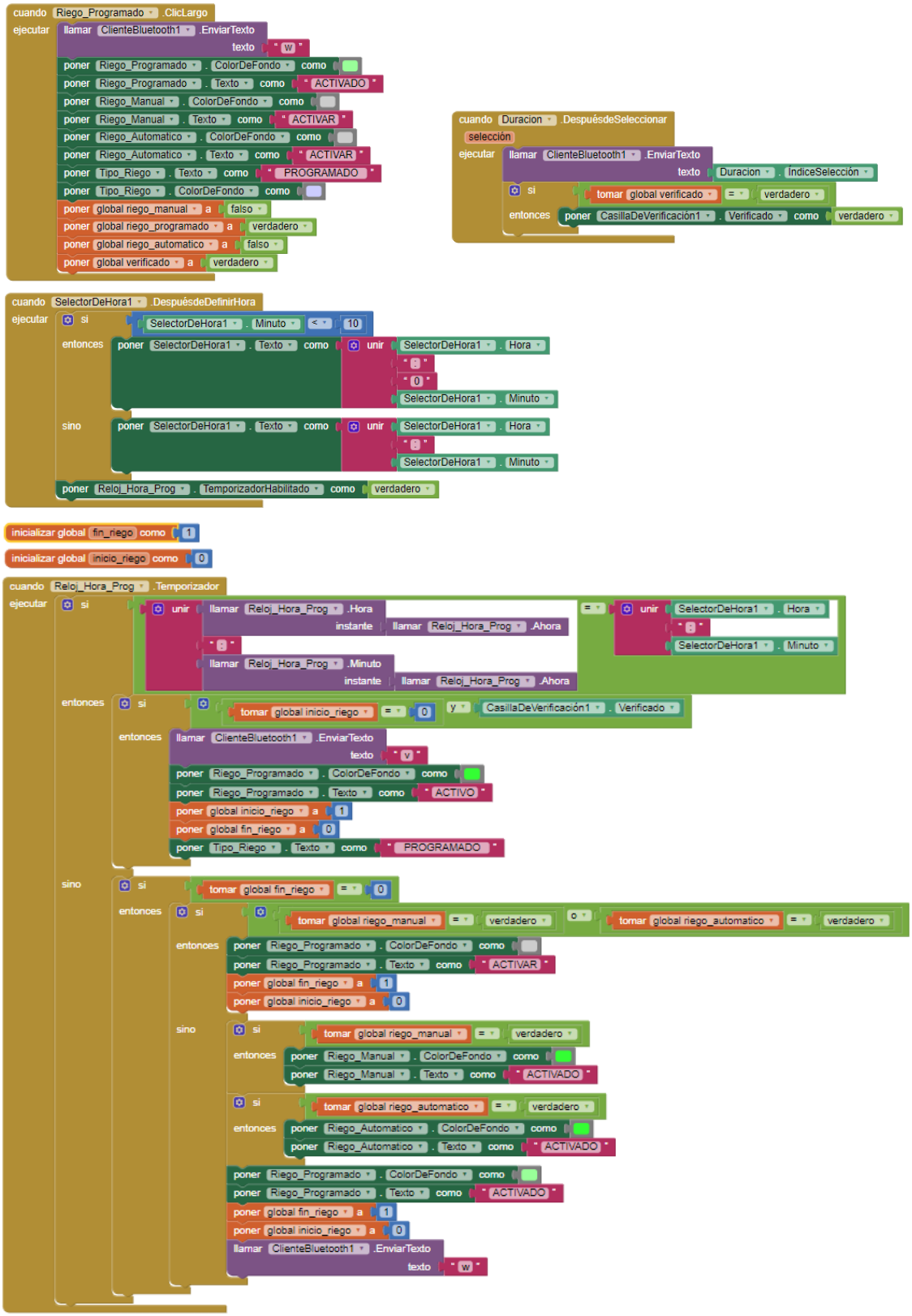


Figura 6.3.4. Aplicación Móvil – Riego Programado

El código más complejo, es el del riego programado, ya que se deben tener en cuenta muchas variables, como la hora a la que establece el inicio del riego, la duración y en qué momento se detiene este riego para dar paso a otro modo de funcionamiento. Lo primero de todo es comparar en todo momento la hora actual con la hora a la que está programado el comienzo del riego. Si esto se cumple y este tipo de riego esta activado se manda una señal al Arduino y el riego comenzará y durará hasta que la duración de este llegue a su fin. Mediante el código anterior queda establecida la hora mediante un selector y el tiempo mediante una lista desplegable. También podemos ver activación y desactivación de señales y variables para llevar el control del programa y acciones de representación de condiciones.

RIEGO AUTOMÁTICO

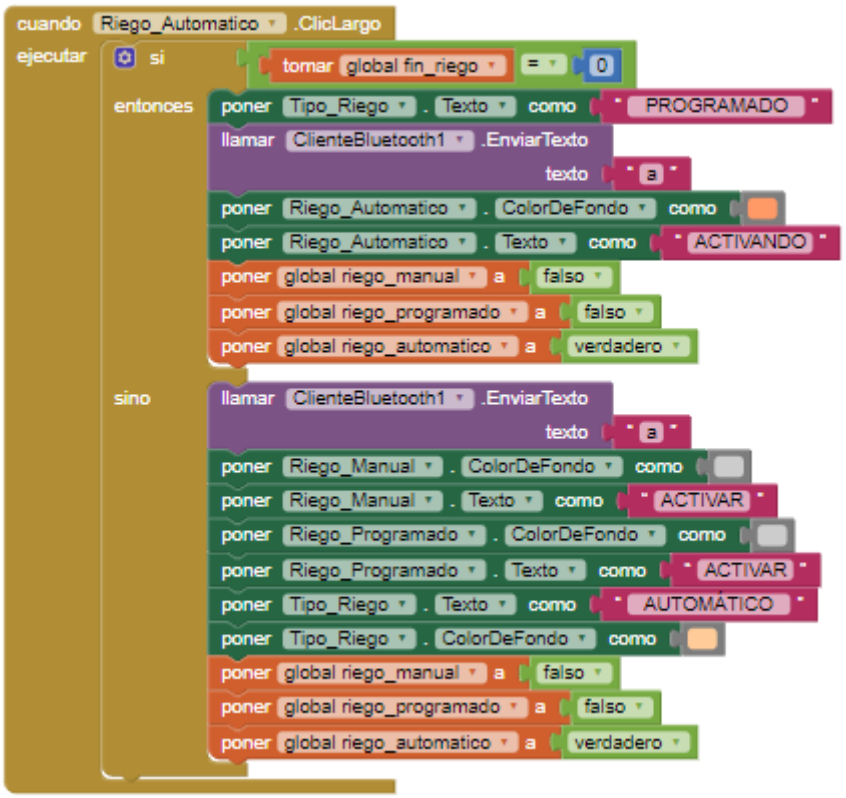


Figura 6.3.5. Aplicación Móvil – Riego Automático

Y el riego automático, es el más sencillo de todos, se activa mediante un botón y su control lo llevará a cabo la placa de Arduino.

Para cambiar de tipo de riego de necesario dejar pulsado el botón de activación durante dos segundos para así tener una confirmación de que se desea cambiar y no se pulse por error. Siempre debe de haber un riego seleccionado por defecto, excepto la primera vez que se inicia la aplicación.

CONCLUSIONES

Se ha construido un control de riego que puede funcionar tanto de manera manual como de forma autónoma manejándose a través de una aplicación móvil, siendo esta última una de las ideas principales del proyecto.

Además, se ha conseguido profundizar en el estudio y conocimiento de las aplicaciones en dispositivos Android a través de la herramienta, MIT App Inventor, la cual proporciona un gran número de posibilidades y oportunidades de desarrollo de proyectos en diferentes ámbitos.

Cabe destacar, también, que al utilizar circuitos auxiliares comerciales, tales como diferentes sensores y sistemas de alimentación, se ha conseguido crear un sistema flexible y modular, capaz de adaptarse a las exigencias requeridas por todos sus posibles usuarios.

Como he comentado anteriormente, en este proyecto se ha utilizado la herramienta MIT APP INVENTOR (herramienta de desarrollo de software online, vía navegador web) con la que he podido ampliar conocimientos y aptitudes y así complementar los contenidos aprendidos durante el grado. Esto lo he podido llevar a cabo investigando vía internet, haciendo un curso de introducción que facilita la propia web y por el método de prueba o error, es decir, cometiendo fallos y solucionándolos.

Como futuras mejoras y ampliaciones, cabe destacar la posibilidad de diseñar un sistema para explotaciones de mayor tamaño, utilizando la energía solar como fuente de alimentación para así disponer de un desarrollo más sostenible, y, a su vez, sería interesante estudiar la posibilidad de tener un control mediante vía telemática o telefonía móvil, para poder controlar la instalación de forma completamente remota.

BIBLIOGRAFÍA

[1] Paula Zapatera Rodríguez. 2022. *Sistema de riego automatizado para pequeñas instalaciones.* [Trabajo Fin de Grado, Universidad Valladolid]

[2] Significados. (s/f). Marco teórico: ejemplos y definición, Recuperado el 18 de septiembre de 2023,

https://www.significados.com/marco-teorico-ejemplos/#:~:text=El%20marco%20te%C3%B3rico%20es%20la,son%20claves%20para%20nuestro%20trabajo.

[3] Grupo Chamartín. (s/f). La intrincada historia de los sistemas de riego. Recuperado el 18 de septiembre de 2023, de https://grupocham

[4] Storyboard That. (s/f). Innovaciones: Irrigación. Recuperado el 18 de septiembre de 2023, de

https://www.storyboardthat.com/es/innovations/irrigaci%C3%B3n

[5] Ecoosfera. (s/f). 3 sistemas de riego de la antigüedad que resuelven los problemas de la actualidad. Recuperado el 18 de septiembre de 2023, de

https://ecoosfera.com/medio-ambiente/3-sistemas-de-riego-de-la-antiguedad-que-resuelven-los-problemas-de-la-actualidad/?utm_content=cmp-true

[6] Código Puebla. publicado en la revista Sciences et Avenir Hors-serie n ° 198, de julio a agosto de 2019. Baolis de la India: pozos comunitarios como templos de piedra. Recuperado el 21 de septiembre de 2023, de

https://www.codigopuebla.com/baolis-de-la-india-pozos-comunitarios-como-templos-de-piedra/

[7] National Geographic. Año 1985. Acueductos: obra maestra de la ingeniería romana. Recuperado el 21 de septiembre de 2023, de

https://historia.nationalgeographic.com.es/a/acueductos-obra-maestra-ingenieria-romana_19151

[8] CITI. (s/f). Los sistemas de riego más recomendados para cada tipo de cultivo. Recuperado el 22 de septiembre de 2023, de

https://citi-sa.com/los-sistemas-de-riego-mas-recomendados-para-cada-tipo-de-cultivo/

[9] iAgua. (s/f). ¿Cuántos tipos de riego hay? Recuperado el 22 de septiembre de 2023, de https://www.iagua.es/respuestas/cuantos-tipos-riego-hay

[10] Fundación Aquae. (s/f). Tipos de riego. Recuperado el 26 de septiembre de 2023, de https://www.fundacionaquae.org/wiki/tipos-de-riego/

[11] Ministerio de Agricultura, Pesca y Alimentación. (s/f). Consulta de páginas sobre material de riego. Recuperado el 26 de septiembre de 2023, de

https://www.mapa.gob.es/es/ministerio/servicios/informacion/plataforma-de-conocimiento-para-el-medio-rural-y-pesquero/observatorio-de-tecnologias-probadas/material-de-riego/consulta-paginas.aspx

[12] Agrohuerto. (s/f). Riego por microaspersión. Recuperado el 26 de septiembre de 2023, de https://www.agrohuerto.com/riego-por-microaspersion/

[13] PasionElectronica. (s/f). Pines y conectores de Arduino Uno. Recuperado el 30 de septiembre de 2023, de

https://pasionelectronica.com/pines-y-conectores-de-arduino-uno/

[14] Proyecto Arduino. (s/f). Arduino Uno R3. Recuperado el 30 de septiembre de 2023, de https://proyectoarduino.com/arduino-uno-r3/

[15] HetPro. (s/f). Pines Arduino. Recuperado el 30 de septiembre de 2023, de https://hetpro-store.com/TUTORIALES/pines-arduino/

[16] Programar Fácil. (s/f). Arduino Uno R3. Recuperado el 30 de septiembre de 2023, de https://programarfacil.com/blog/arduino-blog/arduino-uno-r3/

[17] aylamp Mechatronics. (s/f). Sensor de humedad de suelo capacitivo V1. Recuperado el 15 de julio de 2023, de

https://naylampmechatronics.com/sensores-temperatura-y-humedad/538-sensor-de-humedad-de-suelo-capacitivo-v1.html

[18] Naylamp Mechatronics. (s/f). Tutorial: Sensor digital de temperatura DS18B20. Recuperado el 15 de julio de 2023, de https://naylampmechatronics.com/blog/46_tutorial-sensor-digital-de-temperatura-ds18b20.html

[19] HetPro. (s/f). Tutorial: Comunicación I2C. Recuperado el 15 de julio de 2023, de https://hetpro-store.com/TUTORIALES/i2c/

[20] Control Automático Educación. Sergio Andrés Castaño(s/f). Comunicación I2C. Recuperado el 15 de julio de 2023, de

https://controlautomaticoeducacion.com/microcontroladores-pic/comunicacion-i2c/#google_vignette

[21] Llamas, L. (s/f). Arduino I2C. Recuperado el 15 de julio de 2023, de https://www.luisllamas.es/arduino-i2c/

[22] Arduino Forum. https://forum.arduino.cc/

[23] MIT App Inventor. (s/f). Get Started. Recuperado el 5 de octubre de 2023, de https://appinventor.mit.edu/explore/get-started

[24] MIT App Inventor Community. (s/f). MIT App Inventor Help. Recuperado el 5 de octubre de 2023, de

https://community.appinventor.mit.edu/c/mit-app-inventor-help/5

ANEXOS

A continuación, se adjuntas las características más importantes de cada componente utilizado, extraídas de sus correspondientes *datasheets*. Estas hojas de características se incluyen completas en la carpeta de anejos junto con el código de programa de Arduino y el de la aplicación móvil.

Arduino Uno R3

Sensor de humedad

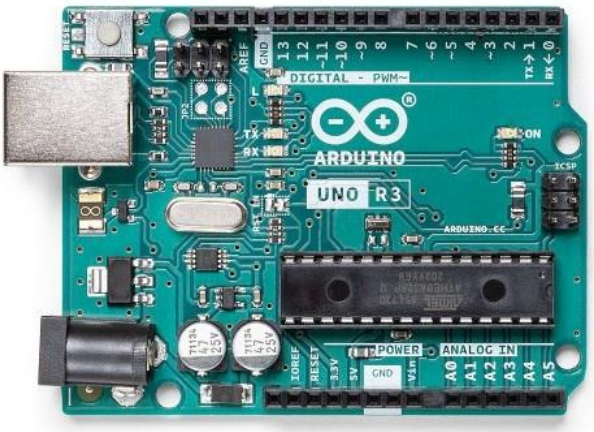
Sensor de temperatura

Display LCD

Bombas de agua

Modulo Bluetooth

Product Reference Manual
SKU: A000066



Description

The Arduino UNO R3 is the perfect board to get familiar with electronics and coding. This versatile development board is equipped with the well-known ATmega328P and the ATMega 16U2 Processor.
This board will give you a great first experience within the world of Arduino.

Target areas:

Maker, introduction, industries

Features

- **ATMega328P** Processor
  - **Memory**
    - AVR CPU at up to 16 MHz
    - 32KB Flash
    - 2KB SRAM
    - 1KB EEPROM
  - **Security**
    - Power On Reset (POR)
    - Brown Out Detection (BOD)
  - **Peripherals**
    - 2x 8-bit Timer/Counter with a dedicated period register and compare channels
    - 1x 16-bit Timer/Counter with a dedicated period register, input capture and compare channels
    - 1x USART with fractional baud rate generator and start-of-frame detection
    - 1x controller/peripheral Serial Peripheral Interface (SPI)
    - 1x Dual mode controller/peripheral I2C
    - 1x Analog Comparator (AC) with a scalable reference input
    - Watchdog Timer with separate on-chip oscillator
    - Six PWM channels
    - Interrupt and wake-up on pin change
  - **ATMega16U2 Processor**
    - 8-bit AVR® RISC-based microcontroller
  - **Memory**
    - 16 KB ISP Flash
    - 512B EEPROM
    - 512B SRAM
    - debugWIRE interface for on-chip debugging and programming
  - **Power**
    - 2.7-5.5 volts

1 The Board

1.1 Application Examples

The UNO board is the flagship product of Arduino. Regardless if you are new to the world of electronics or will use the UNO as a tool for education purposes or industry-related tasks, the UNO is likely to meet your needs.

First entry to electronics: If this is your first project within coding and electronics, get started with our most used and documented board; Arduino UNO. It is equipped with the well-known ATmega328P processor, 14 digital input/output pins, 6 analog inputs, USB connections, ICSP header and reset button. This board includes everything you will need for a great first experience with Arduino.

Industry-standard development board: Using the Arduino UNO R3 board in industries, there are a range of companies using the UNO board as the brain for their PLC's.

Education purposes: Although the UNO R3 board has been with us for about ten years, it is still widely used for various education purposes and scientific projects. The board's high standard and top quality performance makes it a great resource to capture real time from sensors and to trigger complex laboratory equipment to mention a few examples.

1.2 Related Products

- Starter Kit
- Arduino UNO R4 Minima
- Arduino UNO R4 WiFi
- Tinkerkit Braccio Robot

2 Ratings

2.1 Recommended Operating Conditions

Symbol	Description	Min	Max
	Conservative thermal limits for the whole board:	-40 °C (-40°F)	85 °C (185°F)

NOTE: In extreme temperatures, EEPROM, voltage regulator, and the crystal oscillator, might not work as expected.

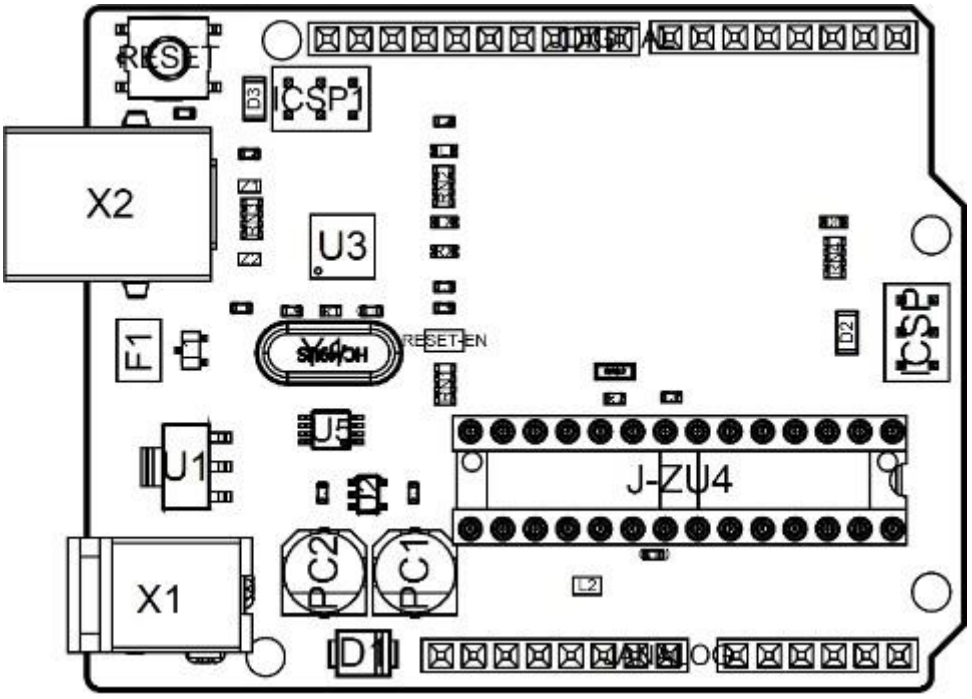
2.2 Power Consumption

Symbol	Description	Min	Typ	Max	Unit
VINMax	Maximum input voltage from VIN pad	6	-	20	V
VUSBMax	Maximum input voltage from USB connector		-	5.5	V
PMax	Maximum Power Consumption	-	-	xx	mA

3 Functional Overview

3.1 Board Topology

Top view



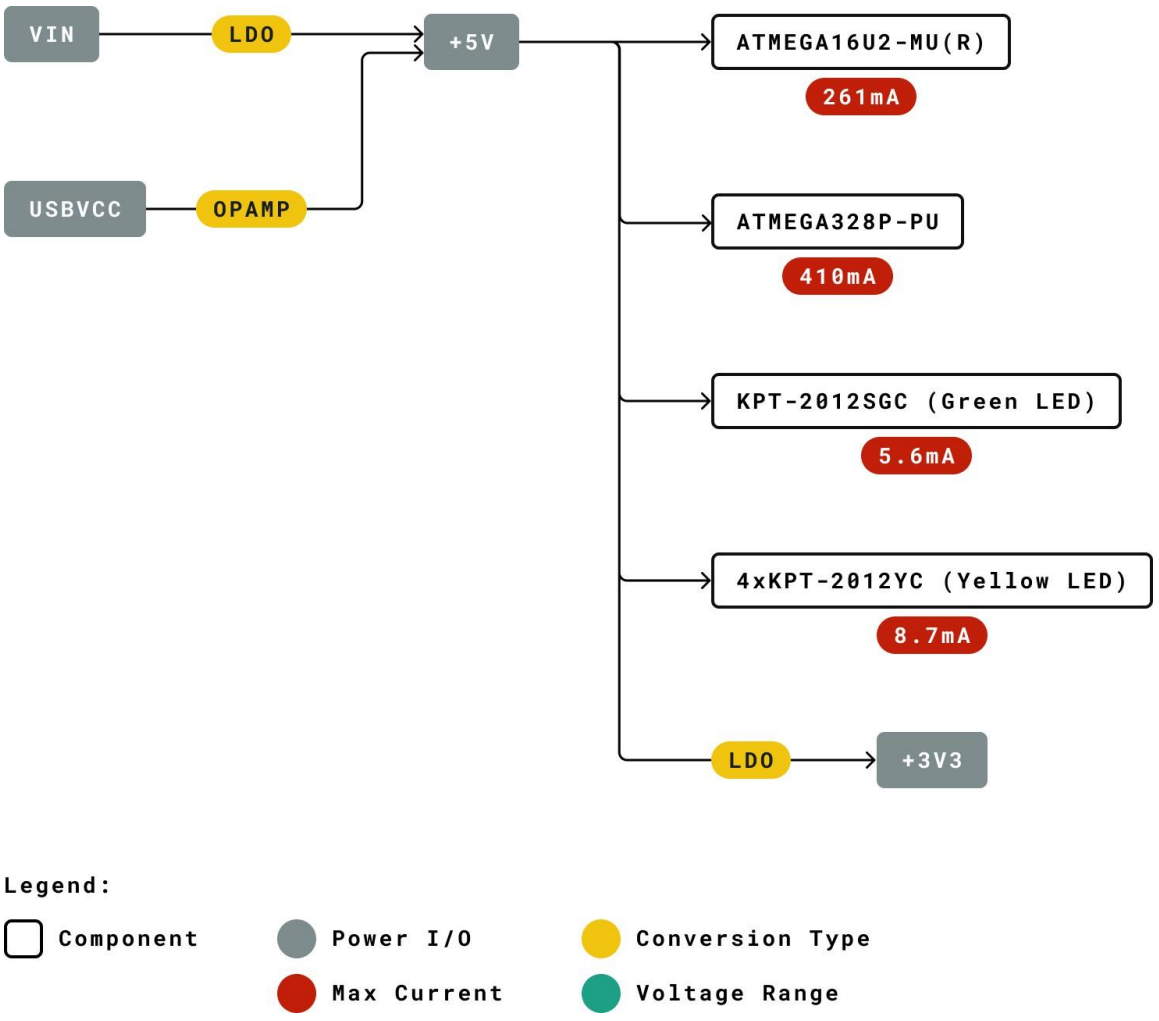
Board topology

Ref.	Description	Ref.	Description
X1	Power jack 2.1x5.5mm	U1	SPX1117M3-L-5 Regulator
X2	USB B Connector	U3	ATMEGA16U2 Module
PC1	EEE-1EA470WP 25V SMD Capacitor	U5	LMV358LIST-A.9 IC
PC2	EEE-1EA470WP 25V SMD Capacitor	F1	Chip Capacitor, High Density
D1	CGRA4007-G Rectifier	ICSP	Pin header connector (through hole 6)
J-ZU4	ATMEGA328P Module	ICSP1	Pin header connector (through hole 6)
Y1	ECS-160-20-4X-DU Oscillator		

3.2 Processor

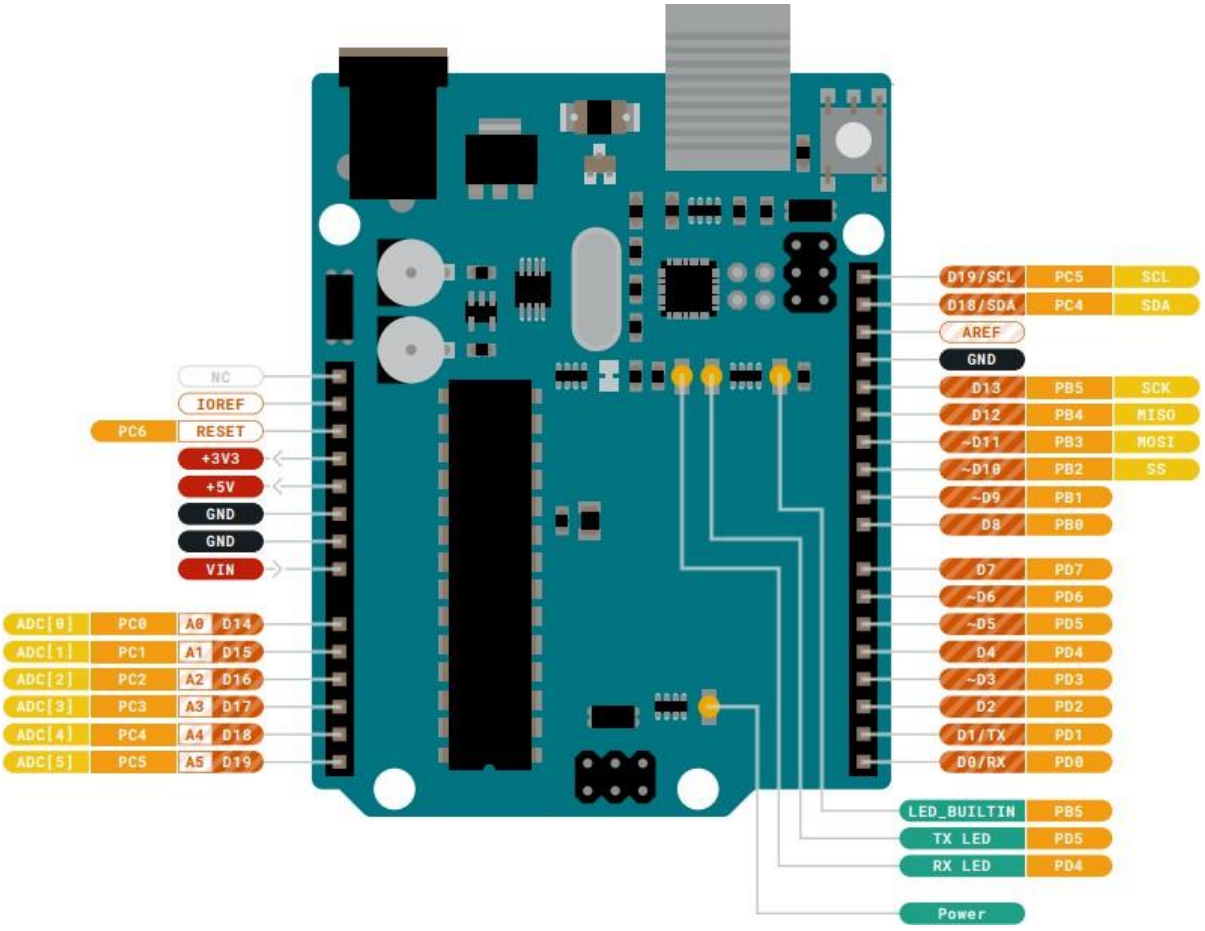
The Main Processor is a ATmega328P running at up to 20 MHz. Most of its pins are connected to the external headers, however some are reserved for internal communication with the USB Bridge coprocessor.

3.3 Power Tree



Power tree

4 Connector Pinouts



Pinout

4.1 JANALOG

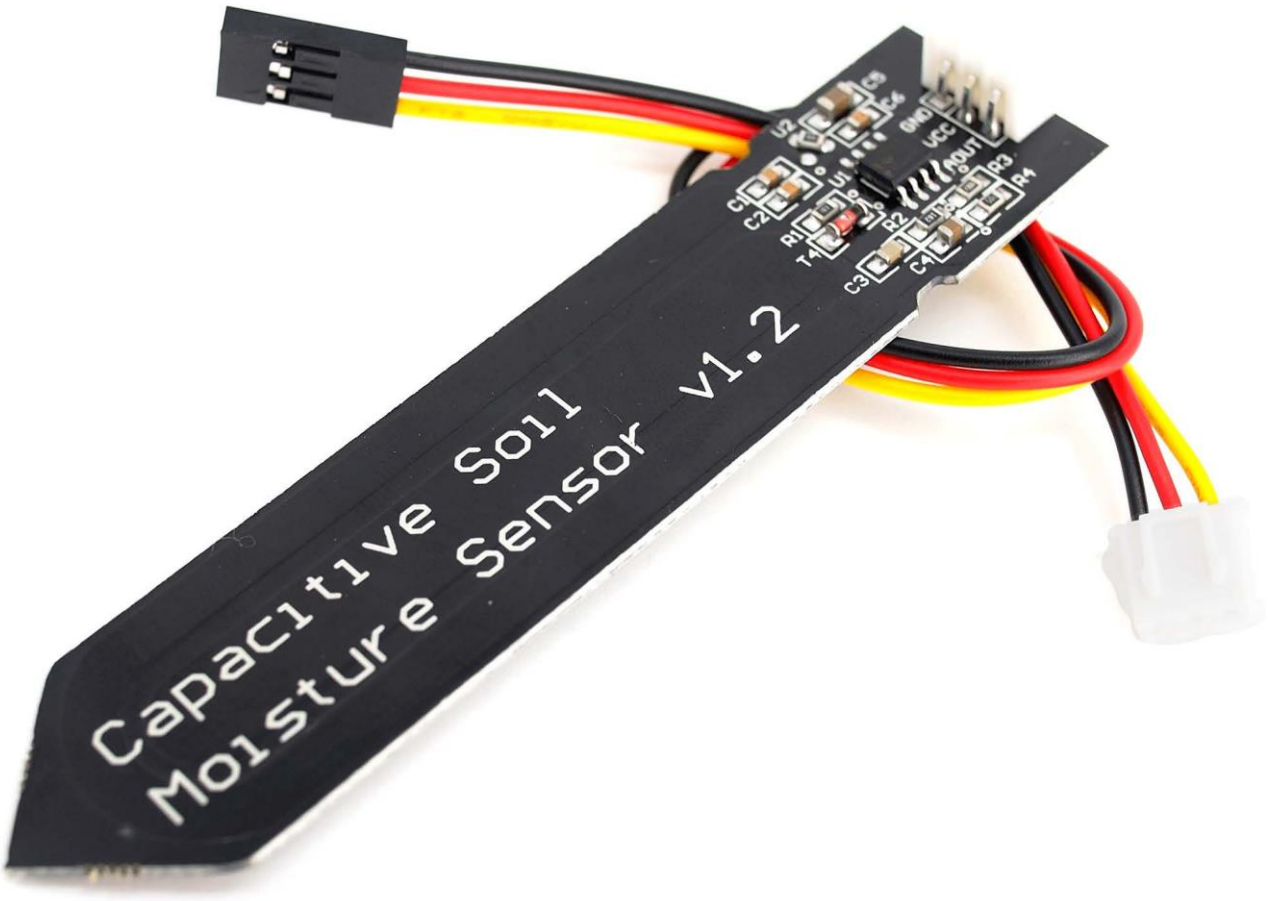
Pin	Function	Type	Description
1	NC	NC	Not connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog/GPIO	Analog input 0 /GPIO
10	A1	Analog/GPIO	Analog input 1 /GPIO
11	A2	Analog/GPIO	Analog input 2 /GPIO
12	A3	Analog/GPIO	Analog input 3 /GPIO
13	A4/SDA	Analog input/I2C	Analog input 4/I2C Data line
14	A5/SCL	Analog input/I2C	Analog input 5/I2C Clock line

4.2 JDIGITAL

Pin	Function	Type	Description
1	D0	Digital/GPIO	Digital pin 0/GPIO
2	D1	Digital/GPIO	Digital pin 1/GPIO
3	D2	Digital/GPIO	Digital pin 2/GPIO
4	D3	Digital/GPIO	Digital pin 3/GPIO
5	D4	Digital/GPIO	Digital pin 4/GPIO
6	D5	Digital/GPIO	Digital pin 5/GPIO
7	D6	Digital/GPIO	Digital pin 6/GPIO
8	D7	Digital/GPIO	Digital pin 7/GPIO
9	D8	Digital/GPIO	Digital pin 8/GPIO
10	D9	Digital/GPIO	Digital pin 9/GPIO
11	SS	Digital	SPI Chip Select
12	MOSI	Digital	SPI1 Main Out Secondary In
13	MISO	Digital	SPI Main In Secondary Out
14	SCK	Digital	SPI serial clock output
15	GND	Power	Ground
16	AREF	Digital	Analog reference voltage
17	A4/SD4	Digital	Analog input 4/I2C Data line (duplicated)
18	A5/SD5	Digital	Analog input 5/I2C Clock line (duplicated)

AZ-Delivery
Ihr Experte für Mikroelektronik!

Hygrometer Modul V1.2
Datenblatt



1. Description

This analog capacitive soil moisture sensor measures soil moisture levels by capacitive sensing, rather than resistive sensing like other types of moisture sensor.
It is made of a corrosion resistant material giving it a long service life.
Insert it into soil and impress your friends with the real-time soil moisture data.

2. Features

- Voltage: 5 V DC
- Interface: PH2.0-3P
- Connecting cable included
- Dimensions: 100 mm x 22.5 mm x 9.5 mm
- Software: Arduino IDE V1.6.5

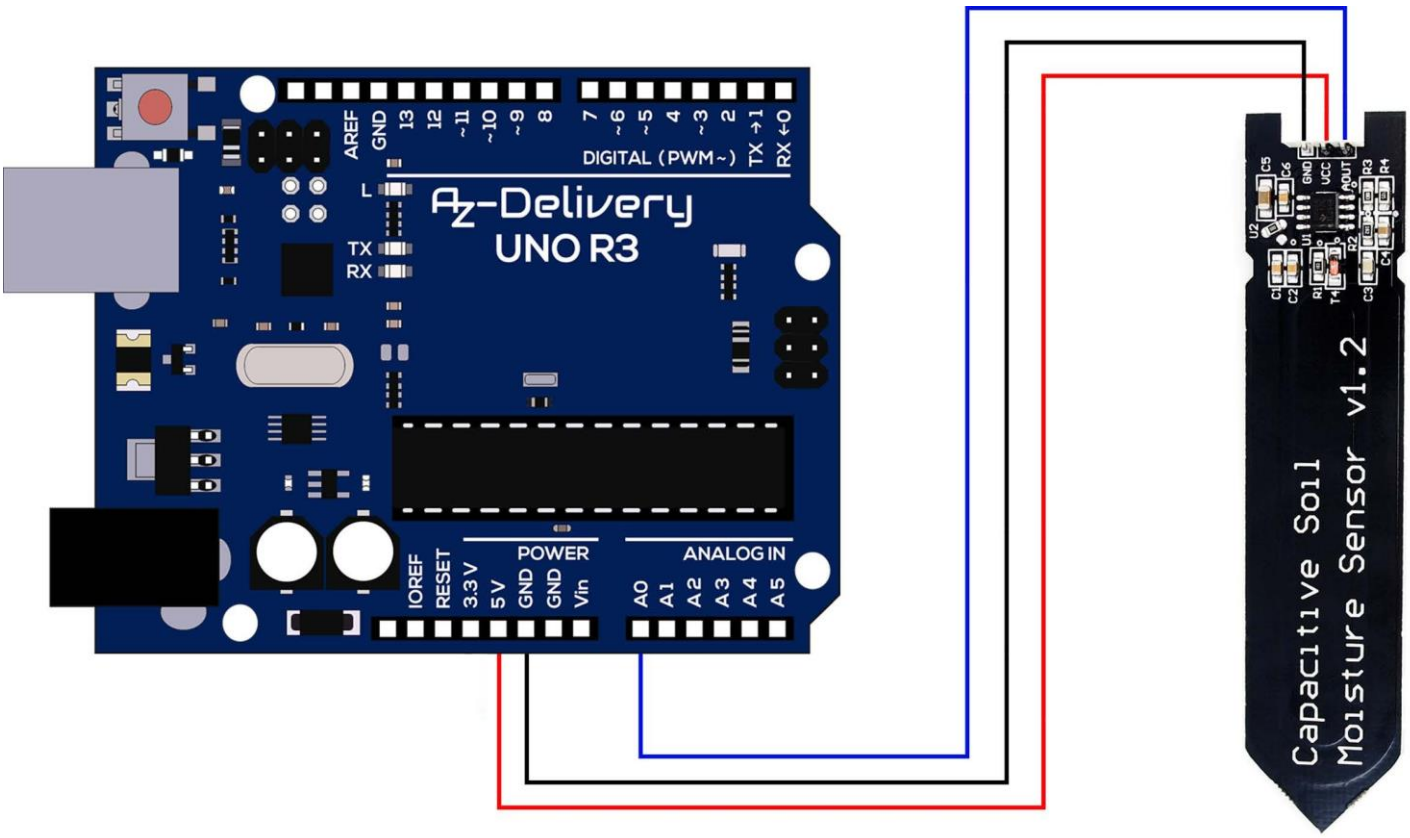
Note: In order to make these sensors work with your Raspberry Pi, an ADC converter is required.

3. Applications

- Garden plants
- Moisture detection
- Intelligent agriculture

Note: Please allow 0-3mm error due to manual measurement.

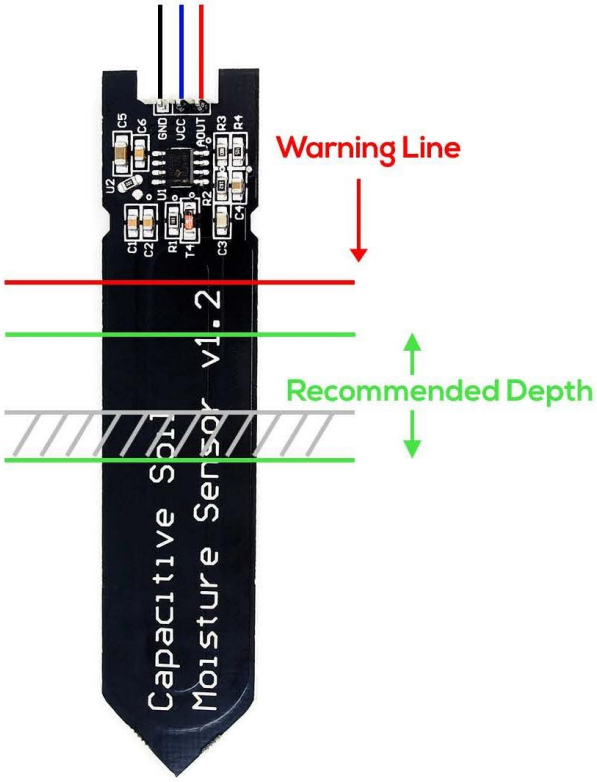
4. Connection Diagram



5. Calibration

Calibration Code

```
void setup() {
Serial.begin(9600); // open serial port, set the baud rate as 9600 bps
}
void loop() {
int val;
val = analogRead(0); //connect sensor to Analog 0
Serial.print(val); //print the value to serial port
delay(100);
}
```



1. Open the serial port monitor and set the baud rate to 9600
2. Record the sensor value when the probe is exposed to the air as "Value 1". This is the boundary value of dry soil “Humidity: 0%RH”
3. Take a cup of water and insert the probe into it no further than the red line in the diagram
4. Record the sensor value when the probe is exposed to the water as "Value 2". This is the boundary value of moist soil “Humidity: 100%RH”

Section Settings

The final output value is affected by probe insertion depth and how tight the soil packed around it is. We regard "value_1" as dry soil and "value_2" as soaked soil. This is the sensor detection range.
For example: Value_1 = 520; Value_2 = 260.
The range will be divided into three sections: dry, wet, water. Their related values are:

- Dry: (520 430]
- Wet: (430 350]
- Water: (350 260]

6. Test Code

```
const int AirValue = 520; //you need to replace this value with Value_1
const int WaterValue = 260; //you need to replace this value with Value_2
int intervals = (AirValue - WaterValue)/3;
int soilMoistureValue = 0;
void setup() {
Serial.begin(9600); // open serial port, set the baud rate to 9600 bps
}
void loop() {
soilMoistureValue = analogRead(A0); //put Sensor insert into soil
if(soilMoistureValue > WaterValue && soilMoistureValue < (WaterValue + int
ervals))
{
Serial.println("Very Wet");
}
else if(soilMoistureValue > (WaterValue + intervals) && soilMoistureValue
< (AirValue - intervals))
{
Serial.println("Wet");
}
else if(soilMoistureValue < AirValue && soilMoistureValue > (AirValue - in
tervals))
{
Serial.println("Dry");
}
delay(100);
}
```

Click here for production status of specific part numbers.

DS18B20

Programmable Resolution 1-Wire Digital Thermometer

General Description

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

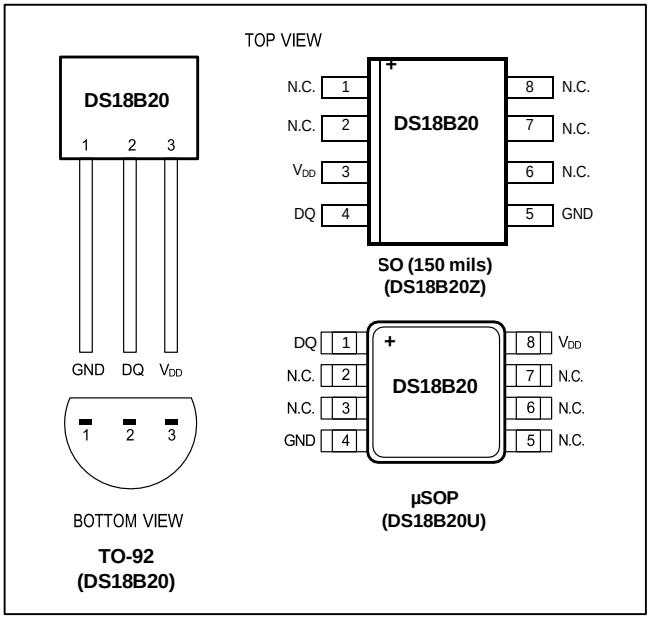
Applications

- Thermostatic Controls
- Industrial Systems
- Consumer Products
- Thermometers
- Thermally Sensitive Systems

Benefits and Features

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages

Pin Configurations



***Ordering Information* appears at end of data sheet.**

1-Wire is a registered trademark of Maxim Integrated Products, Inc.

maxim integrated™

Absolute Maximum Ratings

Voltage Range on Any Pin Relative to Ground .....V to +6.0V
Operating Temperature Range.....-55°C to +125°C
Storage Temperature Range ..... -55°C to +125°C
Solder Temperature .....Refer to the IPC/JEDEC J-STD-020 Specification.

These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

DC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Supply Voltage	V_{DD}	Local power (Note 1)		+3.0		+5.5	V
Pullup Supply Voltage	V_{PU}	Parasite power	(Notes 1, 2)	+3.0		+5.5	V
		Local power		+3.0		V_{DD}	
Thermometer Error	t_{ERR}	-10°C to +85°C	(Note 3)			±0.5	°C
		-30°C to +100°C				±1	
		-55°C to +125°C				±2	
Input Logic-Low	V_{IL}	(Notes 1, 4, 5)		-0.3		+0.8	V
Input Logic-High	V_{IH}	Local power	(Notes 1,6)	+2.2		The lower of 5.5 or V_{DD} + 0.3	V
		Parasite power		+3.0			
Sink Current	I_L	$V_{I/O}$ = 0.4V		4.0			mA
Standby Current	I_{DDS}	(Notes 7, 8)			750	1000	nA
Active Current	I_{DD}	V_{DD} = 5V (Note 9)			1	1.5	mA
DQ Input Current	I_{DQ}	(Note 10)			5		µA
Drift		(Note 11)			±0.2		°C

Note 1: All voltages are referenced to ground.
Note 2: The Pullup Supply Voltage specification assumes that the pullup device is ideal, and therefore the high level of the pullup is equal to V_{PU}. In order to meet the V_{IH} spec of the DS18B20, the actual supply rail for the strong pullup transistor must include margin for the voltage drop across the transistor when it is turned on; thus: $V_{PU\_ACTUAL} = V_{PU\_IDEAL} + V_{TRANSISTOR}$.
Note 3: See typical performance curve in Figure 1. Thermometer Error limits are 3-sigma values.
Note 4: Logic-low voltages are specified at a sink current of 4mA.
Note 5: To guarantee a presence pulse under low voltage parasite power conditions, V_{ILMAX} may have to be reduced to as low as 0.5V.
Note 6: Logic-high voltages are specified at a source current of 1mA.
Note 7: Standby current specified up to +70°C. Standby current typically is 3µA at +125°C.
Note 8: To minimize I_{DDS}, DQ should be within the following ranges: GND ≤ DQ ≤ GND + 0.3V or V_{DD} – 0.3V ≤ DQ ≤ V_{DD}.
Note 9: Active current refers to supply current during active temperature conversions or EEPROM writes.
Note 10: DQ line is high (“high-Z” state).
Note 11: Drift data is based on a 1000-hour stress test at +125°C with V_{DD} = 5.5V.

AC Electrical Characteristics–NV Memory

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
NV Write Cycle Time	t_{WR}			2	10	ms
EEPROM Writes	N_{EEWR}	-55°C to +55°C	50k			writes
EEPROM Data Retention	t_{EEDR}	-55°C to +55°C	10			years

AC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Temperature Conversion Time	t_{CONV}	9-bit resolution	(Note 12)			93.75	ms
		10-bit resolution				187.5	
		11-bit resolution				375	
		12-bit resolution				750	
Time to Strong Pullup On	t_{SPON}	Start convert T command issued				10	µs
Time Slot	t_{SLOT}	(Note 12)		60		120	µs
Recovery Time	t_{REC}	(Note 12)		1			µs
Write 0 Low Time	t_{LOW0}	(Note 12)		60		120	µs
Write 1 Low Time	t_{LOW1}	(Note 12)		1		15	µs
Read Data Valid	t_{RDV}	(Note 12)				15	µs
Reset Time High	t_{RSTH}	(Note 12)		480			µs
Reset Time Low	t_{RSTL}	(Notes 12, 13)		480			µs
Presence-Detect High	t_{PDHIGH}	(Note 12)		15		60	µs
Presence-Detect Low	t_{PDLOW}	(Note 12)		60		240	µs
Capacitance	$C_{IN/OUT}$					25	pF

Note 12: See the timing diagrams in Figure 2.
Note 13: Under parasite power, if t_{RSTL} > 960µs, a power-on reset can occur.

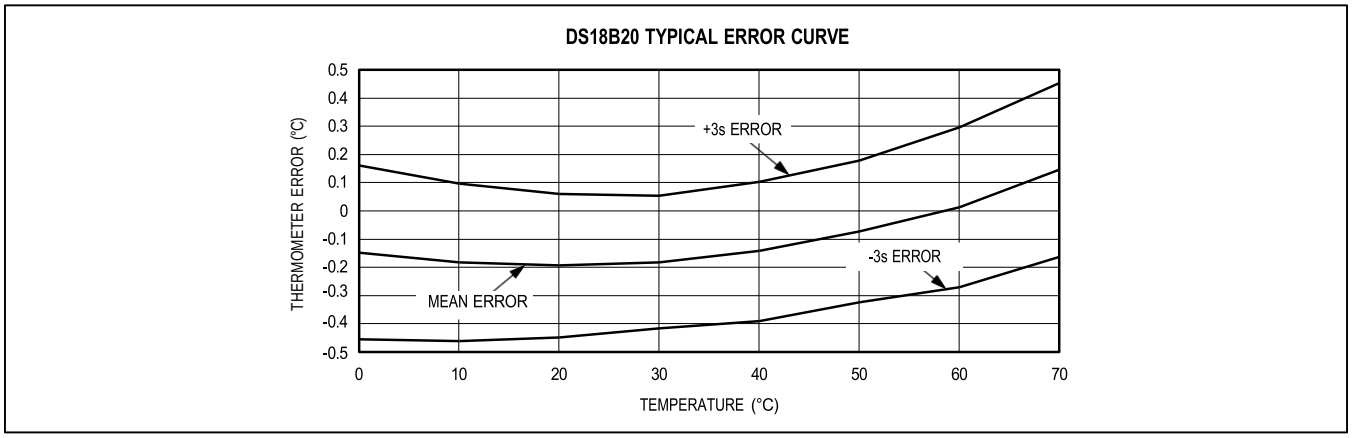


Figure 1. Typical Performance Curve

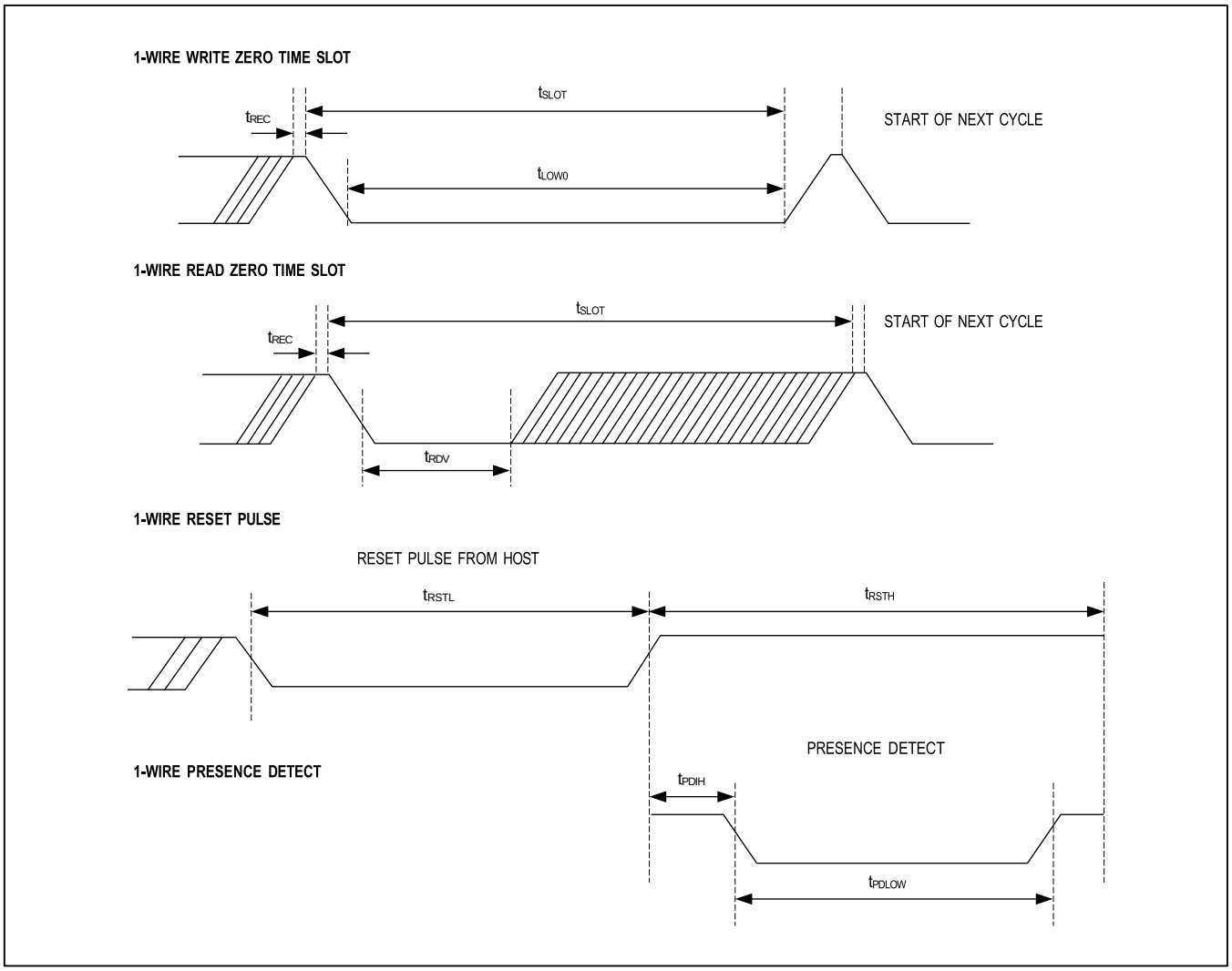


Figure 2. Timing Diagrams

Pin Description

PIN			NAME	FUNCTION
SO	µSOP	TO-92		
1, 2, 6, 7, 8	2, 3, 5, 6, 7	—	N.C.	No Connection
3	8	3	V_{DD}	Optional V_{DD}. V_{DD} must be grounded for operation in parasite power mode.
4	1	2	DQ	Data Input/Output. Open-drain 1-Wire interface pin. Also provides power to the device when used in parasite power mode (see the *Powering the DS18B20* section.)
5	4	1	GND	Ground

Overview

Figure 3 shows a block diagram of the DS18B20, and pin descriptions are given in the *Pin Description* table. The 64-bit ROM stores the device's unique serial code. The scratchpad memory contains the 2-byte temperature register that stores the digital output from the temperature sensor. In addition, the scratchpad provides access to the 1-byte upper and lower alarm trigger registers (T_H and T_L) and the 1-byte configuration register. The configura- tion register allows the user to set the resolution of the temperature-to-digital conversion to 9, 10, 11, or 12 bits. The T_H, T_L, and configuration registers are nonvolatile (EEPROM), so they will retain data when the device is powered down.

The DS18B20 uses Maxim's exclusive 1-Wire bus protocol that implements bus communication using one control signal. The control line requires a weak pullup resistor since all devices are linked to the bus via a 3-state or open-drain port (the DQ pin in the case of the DS18B20). In this bus system, the microprocessor (the master device) identifies and addresses devices on the bus using each device's unique 64-bit code. Because each device has a unique code, the number of devices that can be addressed on one bus is virtually unlimited. The 1-Wire bus protocol, including detailed explanations of the commands and "time slots," is covered in the *1-Wire Bus System* section.

Another feature of the DS18B20 is the ability to operate without an external power supply. Power is instead supplied through the 1-Wire pullup resistor through the DQ pin when the bus is high. The high bus signal also charges an internal capacitor (C_{PP}), which then supplies power to the device when the bus is low. This method of deriving power from the 1-Wire bus is referred to as "parasite power." As an alternative, the DS18B20 may also be powered by an external supply on V_{DD}.

Operation—Measuring Temperature

The core functionality of the DS18B20 is its direct-to-digital temperature sensor. The resolution of the temperature sensor is user-configurable to 9, 10, 11, or 12 bits, corresponding to increments of 0.5°C, 0.25°C, 0.125°C, and 0.0625°C, respectively. The default resolution at power-up is 12-bit. The DS18B20 powers up in a low-power idle state. To initiate a temperature measurement and A-to-D conversion, the master must issue a Convert T [44h] command. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its idle state. If the DS18B20 is powered by an external supply, the master can issue "read time slots" (see the *1-Wire Bus System* section) after the Convert T command and the DS18B20 will respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done. If the DS18B20 is powered with parasite power, this notification technique cannot be used since the bus must be pulled high by a strong pullup during the entire temperature conversion. The bus requirements for parasite power are explained in detail in the *Powering the DS18B20* section.

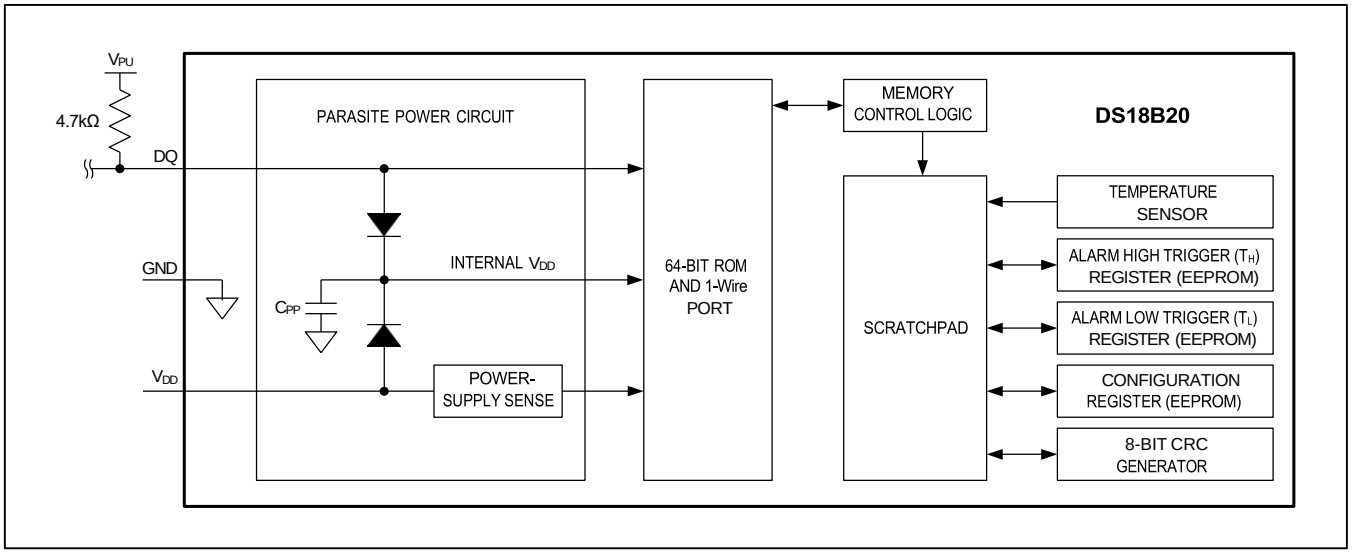


Figure 3. DS18B20 Block Diagram

The DS18B20 output temperature data is calibrated in degrees Celsius; for Fahrenheit applications, a lookup table or conversion routine must be used. The temperature data is stored as a 16-bit sign-extended two's complement number in the temperature register (see Figure 4). The sign bits (S) indicate if the temperature is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. If the DS18B20 is configured for 12-bit resolution, all bits in the temperature register will contain valid data. For 11-bit resolution, bit 0 is undefined. For 10-bit resolution, bits 1 and 0 are undefined, and for 9-bit resolution bits 2, 1, and 0 are undefined. Table 1 gives examples of digital output data and the corresponding temperature reading for 12-bit resolution conversions.

Operation—Alarm Signaling

After the DS18B20 performs a temperature conversion, the temperature value is compared to the user-defined two's complement alarm trigger values stored in the 1-byte T_H and T_L registers (see Figure 5). The sign bit (S) indicates if the value is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. The T_H and T_L registers are nonvolatile (EEPROM) so they will retain data when the device is powered down. T_H and T_L can be accessed through bytes 2 and 3 of the scratchpad as explained in the *Memory* section.

Only bits 11 through 4 of the temperature register are used in the T_H and T_L comparison since T_H and T_L are 8-bit registers. If the measured temperature is lower than

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	**BIT 14**	**BIT 13**	**BIT 12**	**BIT 11**	**BIT 10**	**BIT 9**	**BIT 8**
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

Figure 4. Temperature Register Format

Table 1. Temperature/Data Relationship

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+125	0000 0111 1101 0000	07D0h
+85*	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

**The power-on reset value of the temperature register is +85°C.*

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
S	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Figure 5. T_H and T_L Register Format

or equal to T_L or higher than or equal to T_H, an alarm condition exists and an alarm flag is set inside the DS18B20. This flag is updated after every temperature measurement; therefore, if the alarm condition goes away, the flag will be turned off after the next temperature conversion.

The master device can check the alarm flag status of all DS18B20s on the bus by issuing an Alarm Search [ECh] command. Any DS18B20s with a set alarm flag will respond to the command, so the master can determine exactly which DS18B20s have experienced an alarm condition. If an alarm condition exists and the T_H or T_L settings have changed, another temperature conversion should be done to validate the alarm condition.

Powering the DS18B20

The DS18B20 can be powered by an external supply on the V_{DD} pin, or it can operate in "parasite power" mode, which allows the DS18B20 to function without a local external supply. Parasite power is very useful for applications that require remote temperature sensing or that are very space constrained. Figure 3 shows the DS18B20's parasite-power control circuitry, which "steals" power from the 1-Wire bus via the DQ pin when the bus is high. The stolen charge powers the DS18B20 while the bus is high, and some of the charge is stored on the parasite power capacitor (C_{PP}) to provide power when the bus is low. When the DS18B20 is used in parasite power mode, the V_{DD} pin must be connected to ground.

In parasite power mode, the 1-Wire bus and CPP can provide sufficient current to the DS18B20 for most operations as long as the specified timing and voltage requirements are met (see the *DC Electrical Characteristics* and *AC Electrical Characteristics*). However, when the DS18B20 is performing temperature conversions or copying data from the scratchpad memory to EEPROM, the operating current can be as high as 1.5mA. This current can cause an unacceptable voltage drop across the weak 1-Wire pullup resistor and is more current than can be supplied by C_{PP}. To assure that the DS18B20 has sufficient supply current, it is necessary to provide a strong pullup on the 1-Wire bus whenever temperature conversions are tak- ing place or data is being copied from the scratchpad to EEPROM. This can be accomplished by using a MOSFET to pull the bus directly to the rail as shown in Figure 6. The 1-Wire bus must be switched to the strong pullup within 10µs (max) after a Convert T [44h] or Copy Scratchpad [48h] command is issued, and the bus must be held high by the pullup for the duration of the conversion (t_{CONV}) or data transfer (t_{WR} = 10ms). No other activity can take place on the 1-Wire bus while the pullup is enabled.

The DS18B20 can also be powered by the conventional method of connecting an external power supply to the V_{DD} pin, as shown in Figure 7. The advantage of this method is that the MOSFET pullup is not required, and the 1-Wire bus is free to carry other traffic during the tem- perature conversion time.

The use of parasite power is not recommended for temperatures above +100°C since the DS18B20 may not be able to sustain communications due to the higher leakage currents that can exist at these temperatures. For applications in which such temperatures are likely, it is strongly recommended that the DS18B20 be powered by an external power supply.

In some situations the bus master may not know whether the DS18B20s on the bus are parasite powered or powered by external supplies. The master needs this information to determine if the strong bus pullup should be used during temperature conversions. To get this information, the master can issue a Skip ROM [CCh] command followed by a Read Power Supply [B4h] command followed by a "read time slot". During the read time slot, parasite powered DS18B20s will pull the bus low, and externally powered DS18B20s will let the bus remain high. If the bus is pulled low, the master knows that it must supply the strong pullup on the 1-Wire bus during temperature conversions.

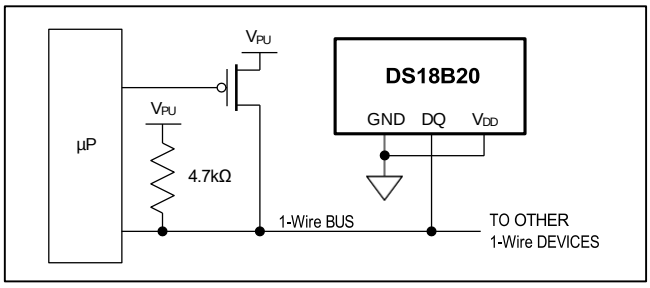


Figure 6. Supplying the Parasite-Powered DS18B20 During Temperature Conversions

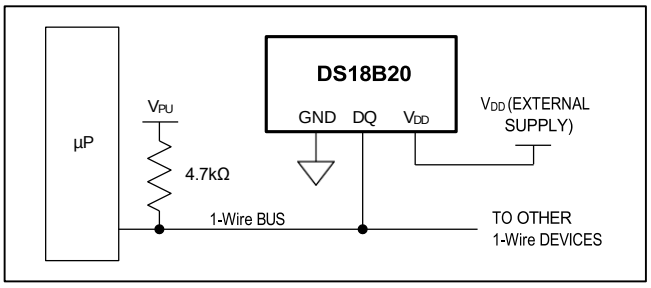


Figure 7. Powering the DS18B20 with an External Supply

1-Wire Bus System

The 1-Wire bus system uses a single bus master to control one or more slave devices. The DS18B20 is always a slave. When there is only one slave on the bus, the system is referred to as a “single-drop” system; the system is “multidrop” if there are multiple slaves on the bus.

All data and commands are transmitted least significant bit first over the 1-Wire bus.

The following discussion of the 1-Wire bus system is broken down into three topics: hardware configuration, transaction sequence, and 1-Wire signaling (signal types and timing).

Hardware Configuration

The 1-Wire bus has by definition only a single data line. Each device (master or slave) interfaces to the data line via an open-drain or 3-state port. This allows each device to “release” the data line when the device is not transmitting data so the bus is available for use by another device. The 1-Wire port of the DS18B20 (the DQ pin) is open drain with an internal circuit equivalent to that shown in Figure 12.

The 1-Wire bus requires an external pullup resistor of approximately 5kΩ; thus, the idle state for the 1-Wire bus is high. If for any reason a transaction needs to be suspended, the bus MUST be left in the idle state if the transaction is to resume. Infinite recovery time can occur between bits so long as the 1-Wire bus is in the inactive (high) state during the recovery period. If the bus is held low for more than 480µs, all components on the bus will be reset.

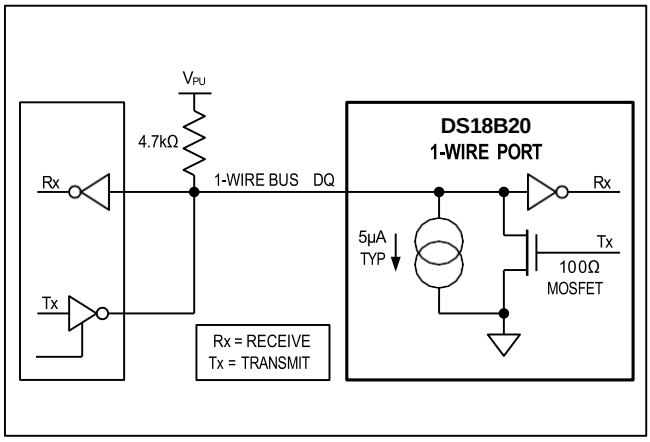


Figure 12. Hardware Configuration

Transaction Sequence

The transaction sequence for accessing the DS18B20 is as follows:

Step 1. Initialization

Step 2. ROM Command (followed by any required data exchange)

Step 3. DS18B20 Function Command (followed by any required data exchange)

It is very important to follow this sequence every time the DS18B20 is accessed, as the DS18B20 will not respond if any steps in the sequence are missing or out of order. Exceptions to this rule are the Search ROM [F0h] and Alarm Search [ECh] commands. After issuing either of these ROM commands, the master must return to Step 1 in the sequence.

Initialization

All transactions on the 1-Wire bus begin with an initialization sequence. The initialization sequence consists of a reset pulse transmitted by the bus master followed by presence pulse(s) transmitted by the slave(s). The presence pulse lets the bus master know that slave devices (such as the DS18B20) are on the bus and are ready to operate. Timing for the reset and presence pulses is detailed in the *1-Wire Signaling* section.

ROM Commands

After the bus master has detected a presence pulse, it can issue a ROM command. These commands operate on the unique 64-bit ROM codes of each slave device and allow the master to single out a specific device if many are present on the 1-Wire bus. These commands also allow the master to determine how many and what types of devices are present on the bus or if any device has experienced an alarm condition. There are five ROM commands, and each command is 8 bits long. The master device must issue an appropriate ROM command before issuing a DS18B20 function command. A flowchart for operation of the ROM commands is shown in Figure 13.

Search Rom [F0h]

When a system is initially powered up, the master must identify the ROM codes of all slave devices on the bus, which allows the master to determine the number of slaves and their device types. The master learns the ROM codes through a process of elimination that requires the master to perform a Search ROM cycle (i.e., Search ROM command followed by data exchange) as many times as necessary to identify all of the slave devices.

If there is only one slave on the bus, the simpler Read ROM [33h] command can be used in place of the Search ROM process. For a detailed explanation of the Search ROM procedure, refer to *Application Note 937: Book of iButton® Standards*. After every Search ROM cycle, the bus master must return to Step 1 (Initialization) in the transaction sequence.

Read Rom [33h]

This command can only be used when there is one slave on the bus. It allows the bus master to read the slave's 64-bit ROM code without using the Search ROM proce- dure. If this command is used when there is more than one slave present on the bus, a data collision will occur when all the slaves attempt to respond at the same time.

Match Rom [55H]

The match ROM command followed by a 64-bit ROM code sequence allows the bus master to address a specific slave device on a multidrop or single-drop bus. Only the slave that exactly matches the 64-bit ROM code sequence will respond to the function command issued by the master; all other slaves on the bus will wait for a reset pulse.

Skip Rom [CCh]

The master can use this command to address all devices on the bus simultaneously without sending out any ROM code information. For example, the master can make all DS18B20s on the bus perform simultaneous temperature conversions by issuing a Skip ROM command followed by a Convert T [44h] command.

Note that the Read Scratchpad [BEh] command can follow the Skip ROM command only if there is a single slave device on the bus. In this case, time is saved by allowing the master to read from the slave without send- ing the device's 64-bit ROM code. A Skip ROM command followed by a Read Scratchpad command will cause a data collision on the bus if there is more than one slave since multiple devices will attempt to transmit data simultaneously.

Alarm Search [ECh]

The operation of this command is identical to the operation of the Search ROM command except that only slaves with a set alarm flag will respond. This command allows the master device to determine if any DS18B20s experienced an alarm condition during the most recent temperature conversion. After every Alarm Search cycle (i.e., Alarm Search command followed by data exchange), the bus master must return to Step 1 (Initialization) in the transaction sequence. See the *Operation—Alarm Signaling* section for an explanation of alarm flag operation.

DS18B20 Function Commands

After the bus master has used a ROM command to address the DS18B20 with which it wishes to communicate, the master can issue one of the DS18B20 function commands. These commands allow the master to write to and read from the DS18B20's scratchpad memory, initiate temperature conversions and determine the power supply mode. The DS18B20 function commands, which are described below, are summarized in Table 3 and illustrated by the flowchart in Figure 14.

Convert T [44h]

This command initiates a single temperature conversion. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its low-power idle state. If the device is being used in parasite power mode, within 10µs (max) after this command is issued the master must enable a strong pullup on the 1-Wire bus for the duration of the conversion (t_{CONV}) as described in the *Powering the DS18B20* section. If the DS18B20 is powered by an external supply, the master can issue read time slots after the Convert T command and the DS18B20 will respond by transmitting a 0 while the temperature conversion is in progress and a 1 when the conversion is done. In parasite power mode this notification technique cannot be used since the bus is pulled high by the strong pullup during the conversion.

Write Scratchpad [4Eh]

This command allows the master to write 3 bytes of data to the DS18B20's scratchpad. The first data byte is written into the T_H register (byte 2 of the scratchpad), the second byte is written into the T_L register (byte 3), and the third byte is written into the configuration register (byte 4). Data must be transmitted least significant bit first. All three bytes MUST be written before the master issues a reset, or the data may be corrupted.

Read Scratchpad [BEh]

This command allows the master to read the contents of the scratchpad. The data transfer starts with the least significant bit of byte 0 and continues through the scratchpad until the 9th byte (byte 8 – CRC) is read. The master may issue a reset to terminate reading at any time if only part of the scratchpad data is needed.

iButton is a registered trademark of Maxim Integrated Products, Inc.

Copy Scratchpad [48h]

This command copies the contents of the scratchpad T_H, T_L and configuration registers (bytes 2, 3 and 4) to EEPROM. If the device is being used in parasite power mode, within 10µs (max) after this command is issued the master must enable a strong pullup on the 1-Wire bus for at least 10ms as described in the *Powering the DS18B20* section.

Recall E^2 [B8h]

This command recalls the alarm trigger values (T_H and T_L) and configuration data from EEPROM and places the data in bytes 2, 3, and 4, respectively, in the scratchpad memory. The master device can issue read time slots following the Recall E^2 command and the DS18B20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done. The recall operation happens automatically at power-up, so valid data is available in the scratchpad as soon as power is applied to the device.

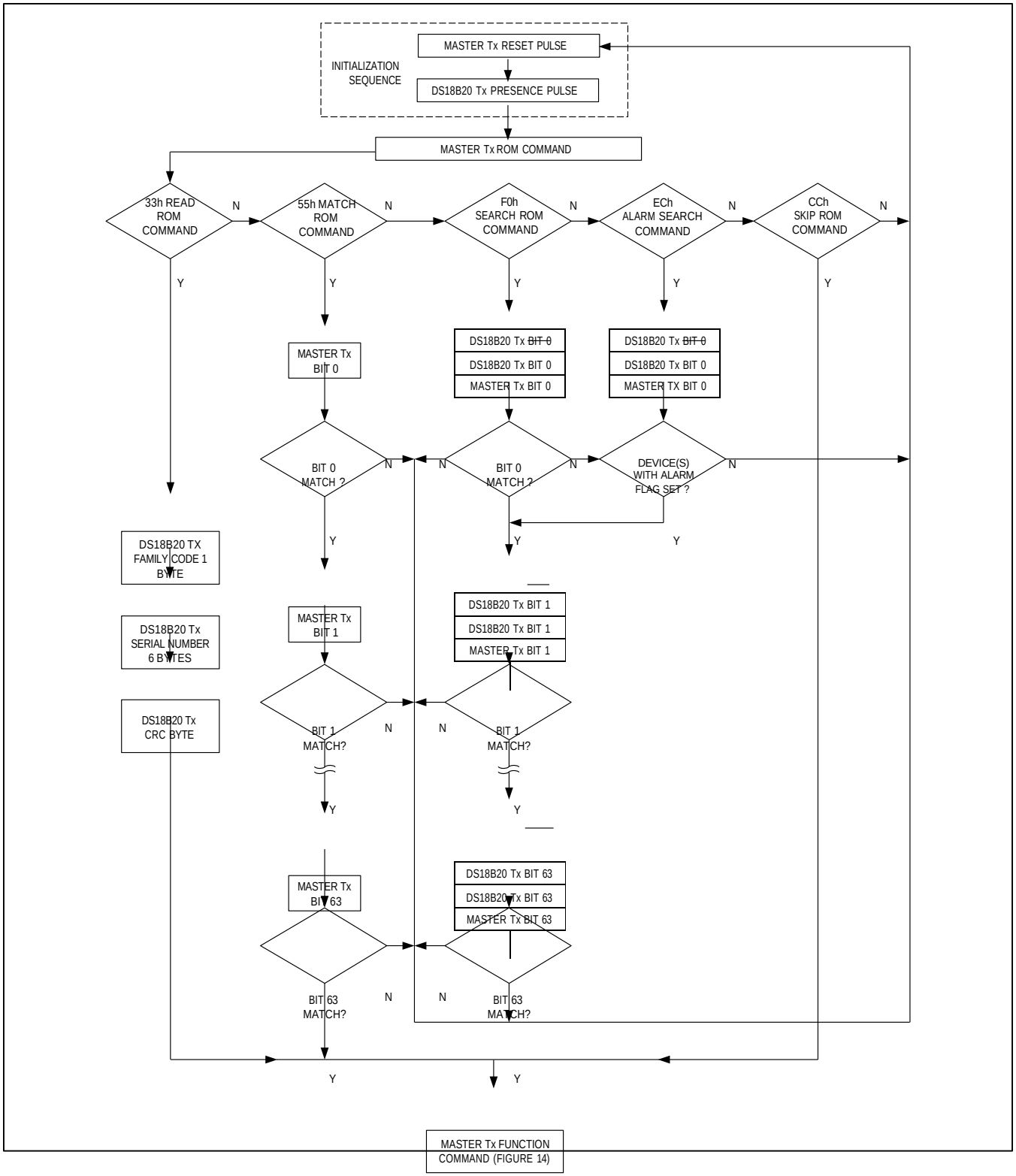
Read Power Supply [B4h]

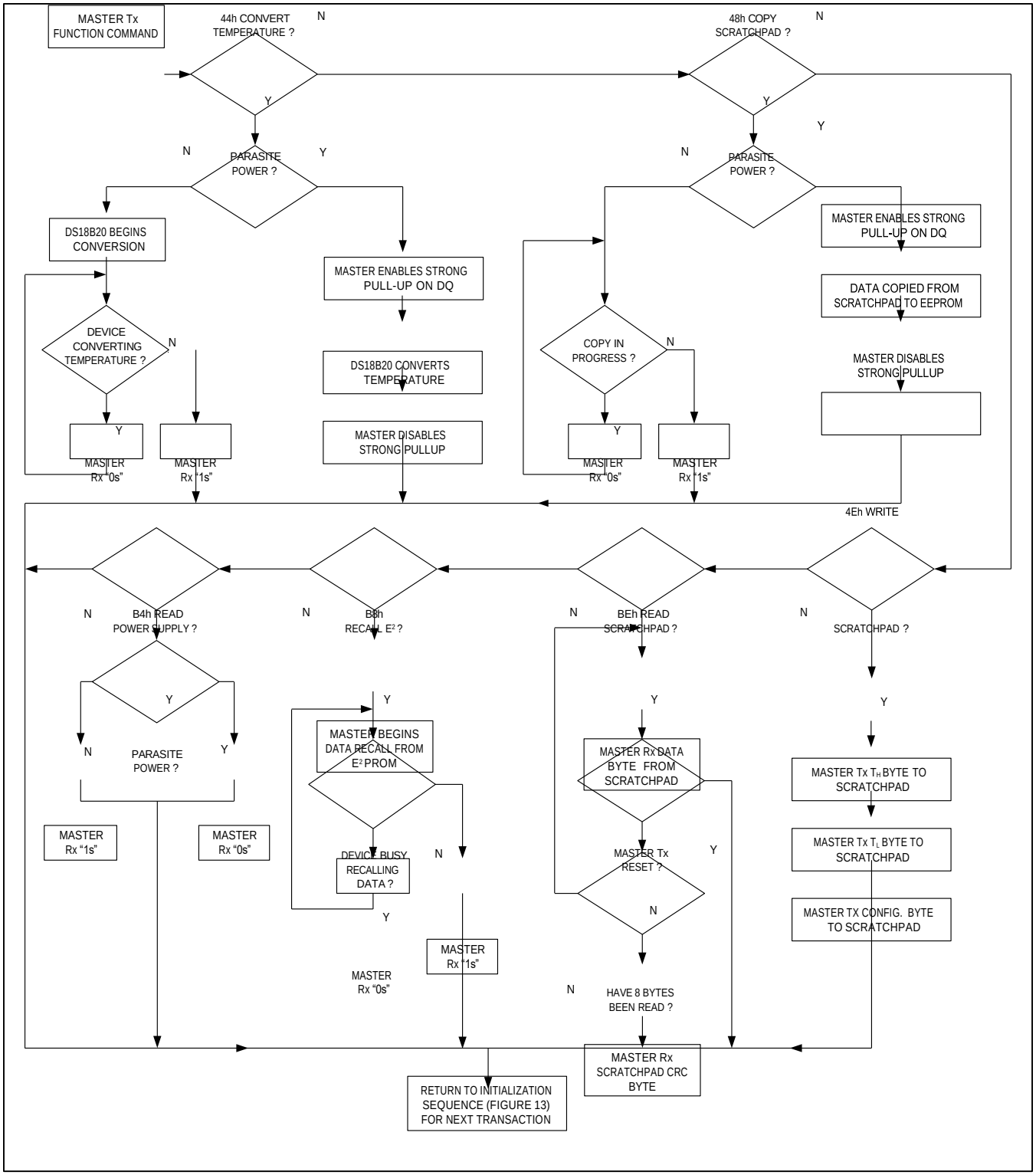
The master device issues this command followed by a read time slot to determine if any DS18B20s on the bus are using parasite power. During the read time slot, parasite powered DS18B20s will pull the bus low, and externally powered DS18B20s will let the bus remain high. See the *Powering the DS18B20* section for usage information for this command.

Table 3. DS18B20 Function Command Set

COMMAND	DESCRIPTION	PROTOCOL	1-Wire BUS ACTIVITY AFTER COMMAND IS ISSUED	NOTES
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	DS18B20 transmits conversion status to master (not applicable for parasite-powered DS18B20s).	1
MEMORY COMMANDS				
Read Scratchpad	Reads the entire scratchpad including the CRC byte.	BEh	DS18B20 transmits up to 9 data bytes to master.	2
Write Scratchpad	Writes data into scratchpad bytes 2, 3, and 4 (T_H, T_L, and configuration registers).	4Eh	Master transmits 3 data bytes to DS18B20.	3
Copy Scratchpad	Copies T_H, T_L, and configuration register data from the scratchpad to EEPROM.	48h	None	1
Recall E^2	Recalls T_H, T_L, and configuration register data from EEPROM to the scratchpad.	B8h	DS18B20 transmits recall status to master.	
Read Power Supply	Signals DS18B20 power supply mode to the master.	B4h	DS18B20 transmits supply status to master.	

Note 1: For parasite-powered DS18B20s, the master must enable a strong pullup on the 1-Wire bus during temperature conversions and copies from the scratchpad to EEPROM. No other bus activity may take place during this time.
Note 2: The master can interrupt the transmission of data at any time by issuing a reset.
Note 3: All three bytes must be written before a reset is issued.


INITIALIZATION SEQUENCE
MASTER Tx RESET PULSE
DS18B20 Tx PRESENCE PULSE
MASTER Tx ROM COMMAND
33h READ ROM COMMAND
55h MATCH ROM COMMAND
F0h SEARCH ROM COMMAND
ECh ALARM SEARCH COMMAND
CCh SKIP ROM COMMAND
N
Y
MASTER Tx BIT 0
DS18B20 Tx BIT 0
MASTER Tx BIT 0
DS18B20 Tx BIT 0
MASTER TX BIT 0
BIT 0 MATCH ?
DEVICE(S) WITH ALARM FLAG SET ?
DS18B20 TX FAMILY CODE 1 BYTE
DS18B20 Tx SERIAL NUMBER 6 BYTES
DS18B20 Tx CRC BYTE
MASTER Tx BIT 1
DS18B20 Tx BIT 1
MASTER Tx BIT 1
BIT 1 MATCH?
MASTER Tx BIT 63
DS18B20 Tx BIT 63
MASTER Tx BIT 63
BIT 63 MATCH?
MASTER Tx FUNCTION COMMAND (FIGURE 14)


MASTER Tx FUNCTION COMMAND
44h CONVERT TEMPERATURE ?
48h COPY SCRATCHPAD ?
PARASITE POWER ?
PARASITE POWER ?
DS18B20 BEGINS CONVERSION
MASTER ENABLES STRONG PULL-UP ON DQ
MASTER ENABLES STRONG PULL-UP ON DQ
DATA COPIED FROM SCRATCHPAD TO EEPROM
DEVICE CONVERTING TEMPERATURE ?
DS18B20 CONVERTS TEMPERATURE
COPY IN PROGRESS ?
MASTER DISABLES STRONG PULLUP
MASTER DISABLES STRONG PULLUP
MASTER Rx "0s"
MASTER Rx "1s"
MASTER Rx "0s"
MASTER Rx "1s"
4Eh WRITE
B4h READ POWER SUPPLY ?
B8h RECALL E² ?
BEh READ SCRATCHPAD ?
SCRATCHPAD ?
PARASITE POWER ?
MASTER BEGINS DATA RECALL FROM E² PROM
MASTER Rx DATA BYTE FROM SCRATCHPAD
MASTER Tx T_H BYTE TO SCRATCHPAD
MASTER Rx "1s"
MASTER Rx "0s"
DEVICE BUSY RECALLING DATA ?
MASTER Tx RESET ?
MASTER Tx T_L BYTE TO SCRATCHPAD
MASTER TX CONFIG. BYTE TO SCRATCHPAD
MASTER Rx "1s"
MASTER Rx "0s"
HAVE 8 BYTES BEEN READ ?
MASTER Rx SCRATCHPAD CRC BYTE
RETURN TO INITIALIZATION SEQUENCE (FIGURE 13) FOR NEXT TRANSACTION

DS18B20 Operation Example 1

In this example there are multiple DS18B20s on the bus and they are using parasite power. The bus master initiates a temperature conversion in a specific DS18B20 and then reads its scratchpad and recalculates the CRC to verify the data.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20s respond with presence pulse.
Tx	55h	Master issues Match ROM command.
Tx	64-bit ROM code	Master sends DS18B20 ROM code.
Tx	44h	Master issues Convert T command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for the duration of the conversion (t_{CONV}).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20s respond with presence pulse.
Tx	55h	Master issues Match ROM command.
Tx	64-bit ROM code	Master sends DS18B20 ROM code.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.

DS18B20 Operation Example 2

In this example there is only one DS18B20 on the bus and it is using parasite power. The master writes to the TH, TL, and configuration registers in the DS18B20 scratchpad and then reads the scratchpad and recalculates the CRC to verify the data. The master then copies the scratchpad contents to EEPROM.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	4Eh	Master issues Write Scratchpad command.
Tx	3 data bytes	Master sends three data bytes to scratchpad (T_H, T_L, and config).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	48h	Master issues Copy Scratchpad command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10ms while copy operation is in progress.

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE	TOP MARK
DS18B20	-55°C to +125°C	3 TO-92	18B20
DS18B20+	-55°C to +125°C	3 TO-92	18B20
DS18B20/T&R	-55°C to +125°C	3 TO-92 (2000 Piece)	18B20
DS18B20+T&R	-55°C to +125°C	3 TO-92 (2000 Piece)	18B20
DS18B20-SL/T&R	-55°C to +125°C	3 TO-92 (2000 Piece)*	18B20
DS18B20-SL+T&R	-55°C to +125°C	3 TO-92 (2000 Piece)*	18B20
DS18B20U	-55°C to +125°C	8 FSOP	18B20
DS18B20U+	-55°C to +125°C	8 FSOP	18B20
DS18B20U/T&R	-55°C to +125°C	8 FSOP (3000 Piece)	18B20
DS18B20U+T&R	-55°C to +125°C	8 FSOP (3000 Piece)	18B20
DS18B20Z	-55°C to +125°C	8 SO	DS18B20
DS18B20Z+	-55°C to +125°C	8 SO	DS18B20
DS18B20Z/T&R	-55°C to +125°C	8 SO (2500 Piece)	DS18B20
DS18B20Z+T&R	-55°C to +125°C	8 SO (2500 Piece)	DS18B20

+Denotes a lead-free package. A "+" will appear on the top mark of lead-free packages.

T&R = Tape and reel.

**TO-92 packages in tape and reel can be ordered with straight or formed leads. Choose "SL" for straight leads. Bulk TO-92 orders are straight leads only.*

Revision History

REVISION DATE	DESCRIPTION	PAGES CHANGED
3/1/07	In the Absolute Maximum Ratings section, removed the reflow oven temperature value of +220°C. Reference to JEDEC specification for reflow remains.	19
10/12/07	In the *Operation—Alarm Signaling* section, added “or equal to” in the description for a TH alarm condition	5
	In the *Memory* section, removed incorrect text describing memory.	7
	In the *Configuration Register* section, removed incorrect text describing configuration register.	8
4/22/08	In the *Ordering Information* table, added TO-92 straight-lead packages and included a note that the TO-92 package in tape and reel can be ordered with either formed or straight leads.	2
1/15	Updated *Benefits and Features* section	1
09/18	Updated *DC Electrical Characteristics* table	2
7/19	Updated Figure 12	10

Specification

For

LCD Module

2004A

TABLE OF CONTENTS

1.0 INTRODUCTION

This USER'S MANUAL is introduced the outside dimensions, optical characteristics, electrical characteristics, interface, controller commands, etc. of the custom design LCD module.

1.1 FEATURE

(1) Display mode: STN POSITIVE, TRANSFLECTIVE, YELLOW-GREEN COLOR

(2) Display format: 20 characters X 4 line

(3) Driving method: 1/16 Duty, 1/5 Bias

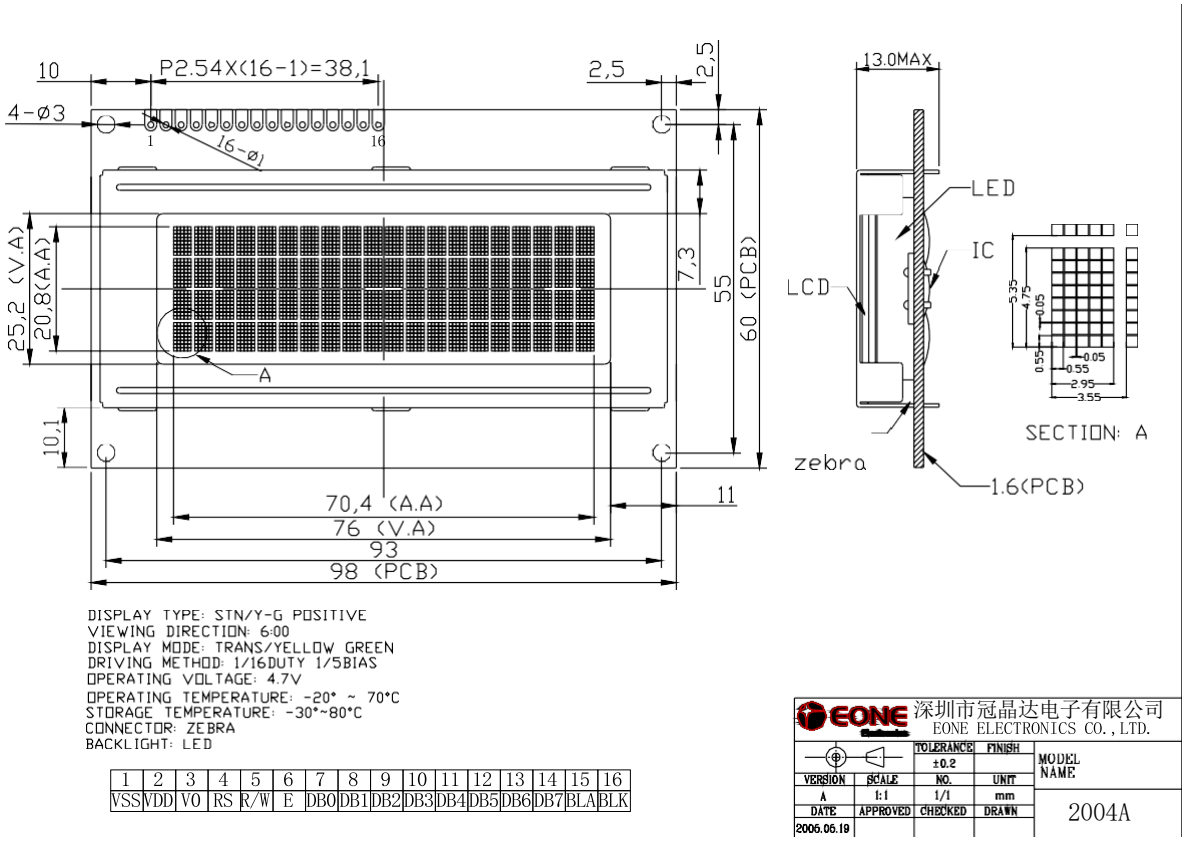
(4) Viewing direction: 6 o'clock

(5) Control IC: SPLC780D

(6) Interface Input Data : 4-Bits or 8-Bits interface available

(7) Back light: LED (Yellow-Green)

2.0 DIMENSION DIAGRAM



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
VSS	VDD	V0	RS	R/W	E	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7	BLA	BLK

3.0 MECHANICAL SPECIFICATIONS

ITEM	STANDARD VALUE	UNIT
DOTS	5X8	characters -
DOT SIZE	0.55X0.55	mm
DOT PITCH	0.60X0.60	mm
MODULE DIMENSION	98.0(W) × 60.0(H) × 1.6(T)	mm
EFFECTIVE DISPLAY AREA	84.0(W) × 31.0(H)MIN	mm

4.0 MAX STANDARD VALUE

ITEM	SYMBOL	MIN.	TYPE	MAX	UNIT
OPERATING TEMPERATURE	Top	-10	25	60	℃
STORAGE TEMPERATURE	Tst	-20	/	70	℃
INPUT VOLTAGE	VI	VSS	/	VDD	V
SUPPLY VOLTAGE FOR LOGIC	VDD-VSS	-0.3	/	7.0	V
SUPPLY VOLTAGE FOR LCD	VDD-V0	VDD-10.0	/	VDD+0.3	V

5.0 ELECTRICAL CHARACTERISTICS

ITEM	SYMBOL	CONDITION	MIN.	TYP.	MAX.	UNIT

SHENZHEN EONT ELECTRONICS CO.,LTD

SUPPLY VOLTAGE FOR LOGIC	V_{DD}-V_{SS}	Ta = 25 ℃	2.7	5.0	5.5	V
SUPPLY VOLTAGE FOR LCD	V_{DD}-V_O (V_{OP})	Ta = 25 ℃	3.0	5.0	10.0	V
INPUT HIGH VOL.	V_{IH}	Ta = 25 ℃	0.7VDD	-	V_{DD}	V
INPUT LOW VOL.	V_{IL}	Ta = 25 ℃	-0.3	-	0.6	V
OUTPUT HIGH VOL.	V_{OH}	Ta = 25 ℃	0.75VDD	-	-	V
OUTPUT LOW VOL.	V_{OL}	Ta = 25 ℃	-	-	0.2VDD	V
SUPPLY CURRENT	I_{DD}	V_{DD} = 3.0V	-	0.1	0.25	mA

6.0 OPTICAL CHARACTERISTICS

No	Item		Symbol	Measurement temperature	MIN.	TYP.	MAX.	Unit
1	Contrast Ratio		Cr	25℃	2.60	3.17		
2	Response Time	Rise time	Tr	25℃	-	-	0.2	us
		Fall time	Tf	25℃	-	-	0.2	us
3	Viewing Angle	6H, Φ=0°	θ 1	25℃	55			Deg.
		12H, Φ=180	θ 2	25℃	0			Deg.
		Φ=90°	θ 3	25℃	45			Deg.
		Φ=270	θ 4	25℃	45			Deg.
4	Frame Frequency			25℃	190	270	350	Hz

6.3. Instruction Table

Instruction	Instruction Code										Description	Execution time (Temp = 25℃)		
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		Fosc= 190KHz	Fosc= 270KHz	Fosc= 350KHz
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM and set DDRAM address to "00H" from AC	2.16ms	1.52ms	1.18ms
Return Home	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	2.16ms	1.52ms	1.18ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Assign cursor moving direction and enable the shift of entire display	53μs	38μs	29μs
Display ON/ OFF Control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor(C), and blinking of cursor(B) on/off control bit.	53μs	38μs	29μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	-	-	Set cursor moving and display shift control bit, and the direction, without changing of DDRAM data.	53μs	38μs	29μs
Function Set	0	0	0	0	1	DL	N	F	-	-	Set interface data length (DL: 8-bit/4-bit), numbers of display line (N: 2-line/1-line) and, display font type (F:5x10 dots/5x8 dots)	53μs	38μs	29μs
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter.	53μs	38μs	29μs
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	53μs	38μs	29μs
Read Busy Flag and Address Counter	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.			
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	53μs	38μs	29μs
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	53μs	38μs	29μs

Note1: "--": don't care

Note2: In the operation condition under -20℃ ~ 75℃, the maximum execution time for majority of instruction sets is 100us, except two instructions, "Clear Display" and "Return Home", in which maximum execution time can take up to 4.1ms.

■ Instruction Description

- **Clear Display**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	0	0	1

Clear all the display data by writing "20H" (space code) to all DDRAM address, and set DDRAM address to "00H" into AC (address counter). Return cursor to the original status, namely, bring the cursor to the left edge on first line of the display. Make entry mode increment (I/D = "1").

- **Return Home**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	0	1	x

Return Home is cursor return home instruction. Set DDRAM address to "00H" into the address counter. Return cursor to its original site and return display to its original status, if shifted. Contents of DDRAM does not change.

- **Entry Mode Set**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	1	I/D	S

Set the moving direction of cursor and display.

- **I/D : Increment / decrement of DDRAM address (cursor or blink)**
  When I/D = "High", cursor/blink moves to right and DDRAM address is increased by 1.
  When I/D = "Low", cursor/blink moves to left and DDRAM address is decreased by 1.
  * CGRAM operates the same as DDRAM, when read from or write to CGRAM.
- **S: Shift of entire display**
  When DDRAM read (CGRAM read/write) operation or S = "Low", shift of entire display is not performed. If S = "High" and DDRAM write operation, shift of entire display is performed according to I/D value (I/D = "1" : shift left, I/D = "0" : shift right).

S	I/D	Description
H	H	Shift the display to the left
H	L	Shift the display to the right

- **Display ON/OFF**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	1	D	C	B

Control display/cursor/blink ON/OFF 1 bit register.

- **D : Display ON/OFF control bit**
  When D = "High", entire display is turned on.
  When D = "Low", display is turned off, but display data is remained in DDRAM.
- **C : Cursor ON/OFF control bit**
  When C = "High", cursor is turned on.
  When C = "Low", cursor is disappeared in current display, but I/D register remains its data.
- **B : Cursor Blink ON/OFF control bit**
  When B = "High", cursor blink is on, that performs alternate between all the high data and display character at the cursor position.
  When B = "Low", blink is off.

- **Cursor or Display Shift**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	1	S/C	R/L	x	x

Without writing or reading of display data, shift right/left cursor position or display. This instruction is used to correct or search display data. During 2-line mode display, cursor moves to the 2nd line after 40th digit of 1st line. Note that display shift is performed simultaneously in all the line. When displayed data is shifted repeatedly, each line shifted individually. When display shift is performed, the contents of address counter are not changed.

S/C	R/L	Description	AC Value
L	L	Shift cursor to the left	AC=AC-1
L	H	Shift cursor to the right	AC=AC+1
H	L	Shift display to the left. Cursor follows the display shift	AC=AC
H	H	Shift display to the right. Cursor follows the display shift	AC=AC

- **Function Set**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	1	DL	N	F	x	x

- **DL : Interface data length control bit**
  When DL = "High", it means 8-bit bus mode with MPU.
  When DL = "Low", it means 4-bit bus mode with MPU. So to speak, DL is a signal to select 8-bit or 4-bit bus mode.
  When 4-bit bus mode, it needs to transfer 4-bit data by two times.
- **N : Display line number control bit**
  When N = "Low", it means 1-line display mode.
  When N = "High", 2-line display mode is set.
- **F : Display font type control bit**
  When F = "Low", it means 5 x 8 dots format display mode
  When F = "High", 5 x11 dots format display mode.

N	F	No. of Display Lines	Character Font	Duty Factor
L	L	1	5x8	1/8
L	H	1	5x11	1/11
H	x	2	5x8	1/16

- **Set CGRAM Address**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

Set CGRAM address to AC.
This instruction makes CGRAM data available from MPU.

- **Set DDRAM Address**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0

Set DDRAM address to AC.
This instruction makes DDRAM data available from MPU.
When 1-line display mode (N = 0), DDRAM address is from "00H" to "4FH".
In 2-line display mode (N = 1), DDRAM address in the 1st line is from "00H" to "27H", and DDRAM address in the 2nd line is from "40H" to "67H".

- **Read Busy Flag and Address**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0

When BF = "High", indicates that the internal operation is being processed.So during this time the next instruction cannot be accepted.

The address Counter (AC) stores DDRAM/CGRAM addresses, transferred from IR.

After writing into (reading from) DDRAM/CGRAM, AC is automatically increased (decreased) by 1.

- **Write Data to CGRAM or DDRAM**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	1	0	D7	D6	D5	D4	D3	D2	D1	D0

Write binary 8-bit data to DDRAM/CGRAM.

The selection of RAM from DDRAM, CGRAM, is set by the previous address set instruction : DDRAM address set, CGRAM address set. RAM set instruction can also determine the AC direction to RAM.

After write operation, the address is automatically increased/decreased by 1, according to the entry mode.

- **Read Data from CGRAM or DDRAM**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	1	1	D7	D6	D5	D4	D3	D2	D1	D0

Read binary 8-bit data from DDRAM/CGRAM.

The selection of RAM is set by the previous address set instruction. If address set instruction of RAM is not performed before this instruction, the data that read first is invalid, because the direction of AC is not determined. If you read RAM data several times without RAM address set instruction before read operation, you can get correct RAM data from the second, but the first data would be incorrect, because there is no time margin to transfer RAM data.

In case of DDRAM read operation, cursor shift instruction plays the same role as DDRAM address set instruction : it also transfer RAM data to output data register. After read operation address counter is automatically increased/decreased by 1 according to the entry mode. After CGRAM read operation, display shift may not be executed correctly.

* In case of RAM write operation, after this AC is increased/decreased by 1 like read operation. In this time, AC indicates the next address position, but you can read only the previous data by read instruction.

10. CHARACTER GENERATOR ROM

10.1. SPLC780D1 – 001A

UNIT Electronics

TIENDA DE COMPONENTES ELECTRÓNICOS

Bomba de agua sumergible 70-120 l/h Micro Motor

Descripción

Está mini bomba es totalmente sumergible y muy funcional para aplicaciones donde requieras extraer agua. Puede funcionar con un rango de tensión de 2.5 a 6 VCD. Permite tener un flujo de hasta de dos litros de agua por minuto (80-120 l/h).

Internamente está conformado por un motor DC de 800mA, además con su hélice te permitirá impulsar el agua que recibe en la entrada, obteniendo a la salida una presión aún mayor.

El motor eléctrico viene incluido en sus terminales unos capacitores cerámicos 104 que los permitirán mejorar la estabilidad de voltaje en el circuito. Pues previene picos y el ruido eléctrico que la fuente de tensión puede generar.

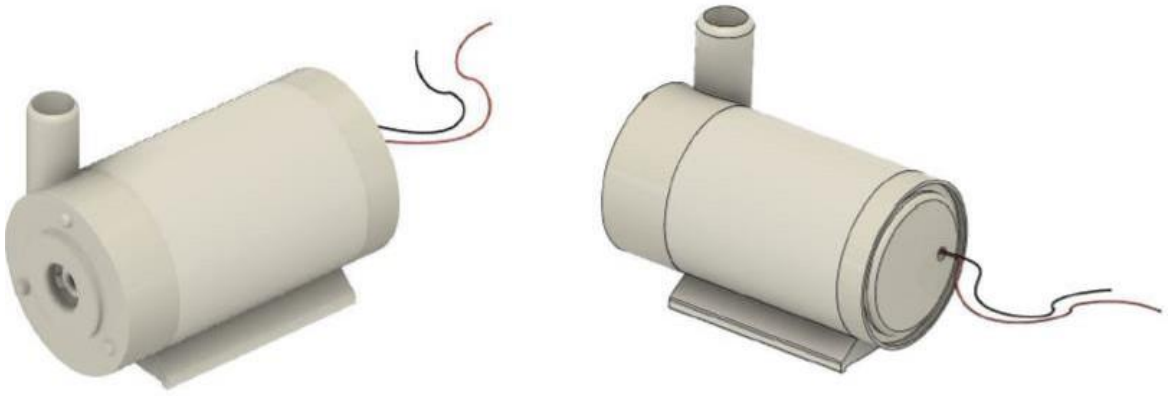


Imagen-.1

Especificaciones técnicas

- Voltaje de trabajo: 2.5 – 6 VCD
- Altura Bombeo máximo: 0.4 – 1.1 metros
- Caudal Bombeo máximo: 80-120 l/H
- Corriente: 800mA
- Potencia: 0.4 -1.5 W
- Horas de trabajo continuo: 500 horas
- Longitud cable: 20 cm
- No cuenta con manguera
- SKU: AR0366
- Color: Amarillento
- Dimensiones: 45x34x23 aproximadamente

Aplicaciones

- Filtros o flujo de acuarios o peceras pequeñas (máx 100 litros)
- Cascadas o fuentes
- Regaderas de pasto

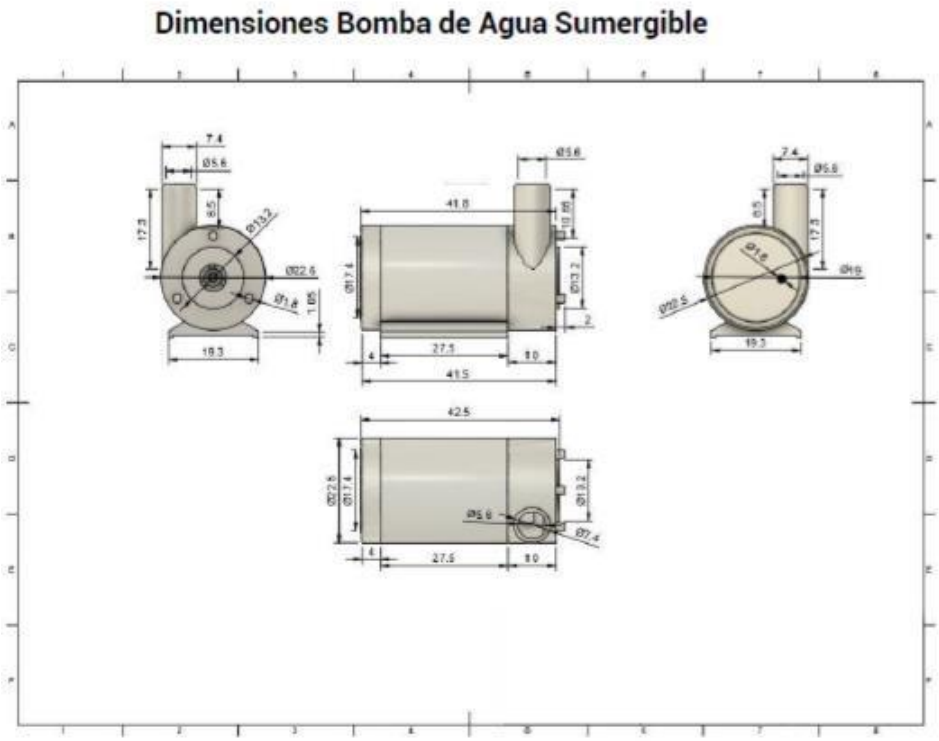


Imagen-.2

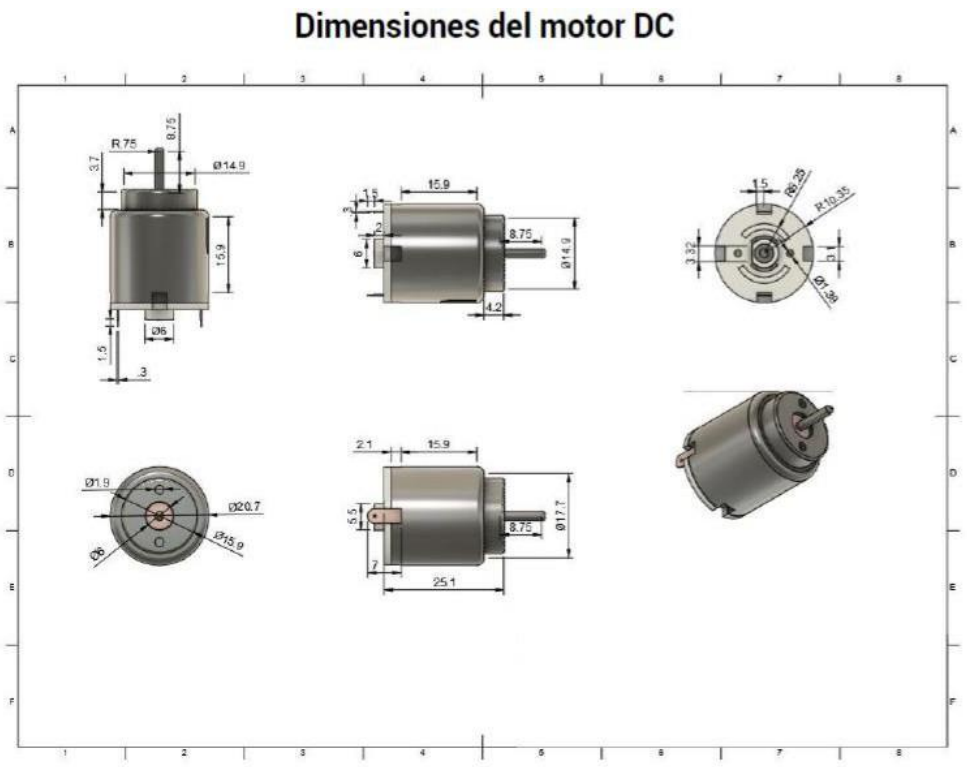


Imagen-.3

Imagen-. 4

Recurso

- Diseño 3D

EGBT-046S

Bluetooth Modules

Wireless UART Cable Replacement

Hardware Manual &
AT Commands Reference Manual Rev. 1r0

EGBT-045MS and EGBT-046S Bluetooth Module are low cost replacements of our now retired EGBC-04 Bluetooth Module. EGBC-04 is an excellent Bluetooth Module, it is fully certified to Bluetooth standards, and is loaded with programmable features users had come to love. There is just one thing that went against it- it is expensive.

It is easy to see why the EGBC-04 cost so much. Firstly, the manufacturer produced these specialty modules in relatively small volume; hence, there is no economy of the scale to speak of. Secondly, certification costs a lot of money; and this cost will have to be added on top of the manufacturing cost. Hence, EGBC-04 ended up costing about 10 times more expensive than its garden variety USB-type Bluetooth dongles cousins.

Fortunately, at least one volume manufacturer have came up with an idea of producing a generic Bluetooth module in large quantity, for sale and distribution to developers who now have to put only the firmware functionalities. This resulted in a huge drop in prices of these specialty Bluetooth modules, benefiting us experimenters and hobbyists.

EGBT-045MS and EGBT-046S are generic Bluetooth Modules loaded with SPP firmware for UART wireless cable replacement functions. The EGBT-045MS can be configured by the user to work either as a master or slave Bluetooth device using a set of AT commands.

EGBT-046S, on the other hand, is permanently programmed as Bluetooth slave device. EGBT-046S, because of its simpler function, is a lot easier to use, and of course, costs less than EGBT-045MS. You can use it straight out of the box as a UART wireless cable replacement, without any need to add set-up codes in your microcontroller application firmware.

The new EGBT-04 Bluetooth module comes in two flavors. The EGBT-046S is permanently configured as a slave device. EGBT-045MS, on the other hand, can be configured by the user to work as a master or slave Bluetooth device.

EGBT-04 modules can be soldered directly on a hi-rel type IC socket to make it easier to work with prototyping platforms, such as breadboards and perforated prototyping boards.

Use the cheaper EGBT-046S if your application will connect to a master Bluetooth device, such as PC or laptops. Use the EGBT-045MS if your application must connect to a slave Bluetooth device, such as with EGBT-046S. Note that EGBT-045MS will work as well as a slave Bluetooth device.

COMMON SPECIFICATIONS

Radio Chip: CSR BC417
Memory: External 8Mbit Flash
Output Power: -4 to +6dbm Class 2
Sensitivity: -80dbm Typical
Bit Rate: EDR, up to 3Mbps
Interface: UART
Antenna: Built-in
Dimension: 27W x 13H mm

Voltage: 3.1 to 4.2VDC
Current: 40mA max

COMMON HARDWARE INTERFACING CONSIDERATIONS

The EGBT-04 module will work with supply voltage of 3.1VDC to 4.2VDC. When supplied with 3.3VDC, it will interface directly with the UART port of any microcontroller chip running at 3.3VDC.

When used with 5V microcontrollers, The TXD output logic swing of the EGBT-04 still falls within the valid 5V TTL range, hence, can be connected directly to the UART RXD of the 5V microcontroller host. EGBT RXD and inputs, however, are not 5V tolerant, and can be damaged by 5V level logic going in. Some level translation circuit must be added to protect the inputs.

A simple diode level translator circuit like the ones shown in Figure 3 and 7 will suffice in most applications. A better alternative is with the use of 5V input tolerant tiny logic chips such as 74LVC1G125 – a single buffer chip housed in smd sot23-5 package.

EGBT-046S PIN CONFIGURATION

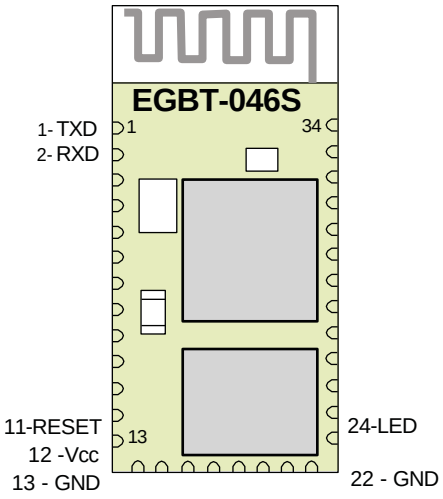


Figure 1. *EGBT-046S Pin Layout*

Table 1. EGBT-046S Pin Description

PIN	ID	DESCRIPTION
1	TXD	UART TXD Output
2	RXD	UART RXD Input
11	RESET	RESET Input
12	Vcc	+3.1 to 4.2VDC Power Input
13	GND	Common Ground
22	GND	Common Ground
24	LED	LED Status Indicator Flashing - Waiting to Connect/Pair Steady ON - Connected/Paired

Note:
All unassigned pins must be left unconnected.

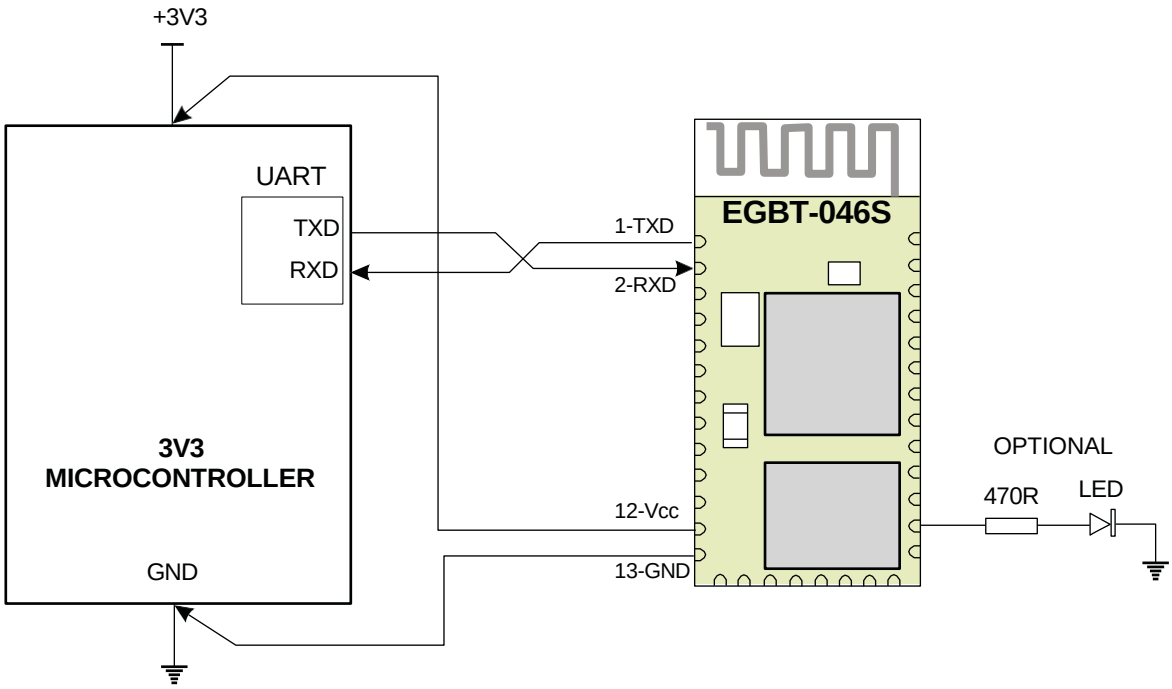


Figure 2. *EGBT-046S wiring example with a 3v3 host microcontroller. The 470R resistor and LED are for status indication, and may be omitted if not needed.*

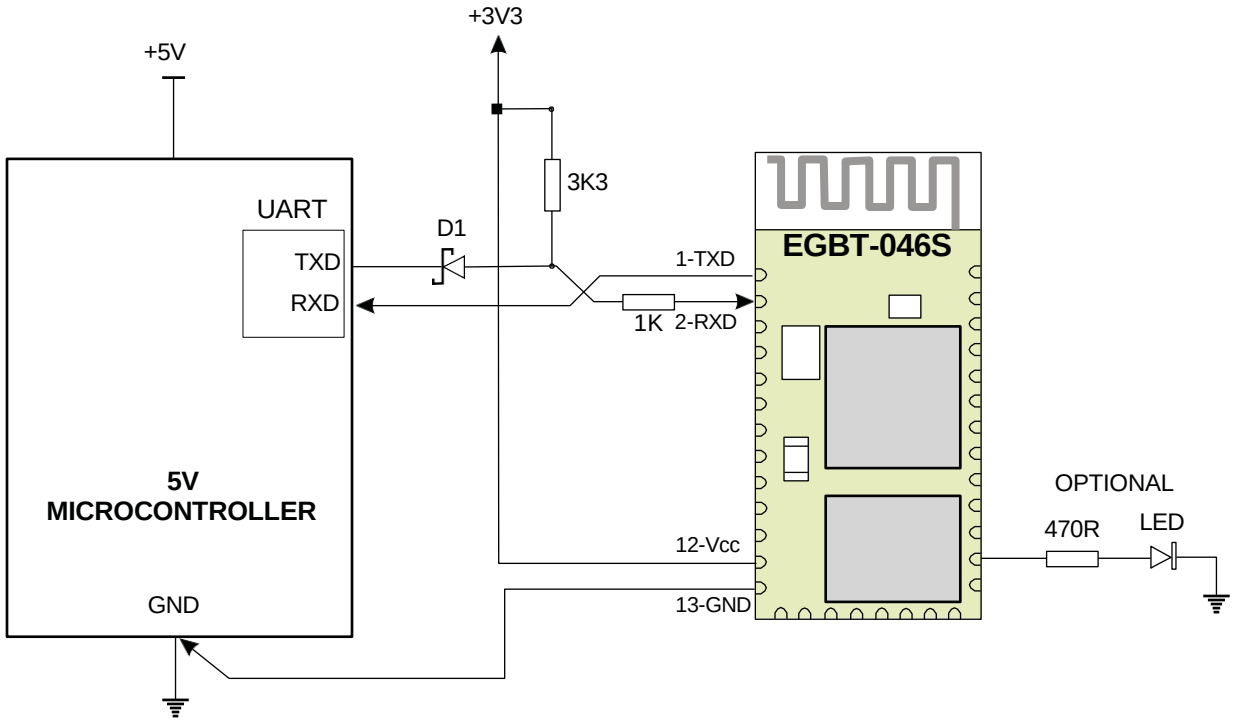


Figure 3. *EGBT-046S RX input is not 5V tolerant. A schottky diode connected as shown will keep 5V voltages out of the Bluetooth module when operated with a 5V host microcontroller.*

PREPARATION FOR USE

The EGBT-046S is permanently configured as a slave Bluetooth device. It works under the following default configuration:

Baud Rate: 9600 bps
Data : 8 bits
Stop Bits: 1 bit
Parity : None
Handshake: None

Passkey: 1234
Device Name: linvor

If the default configuration suits your application, then you can use EGBT-046S immediately. Once it is paired to a master Bluetooth device, its operation becomes transparent to the user. No user code specific to the Bluetooth module is needed at all in the user microcontroller program.

The EGBT-046S automatically sets itself up in Command Mode when it is not remotely connected (paired) to any other Bluetooth device. You can change the Passkey, Device Name, and Baud Rate while the EGBT-046S is in Command Mode by entering a small subset of AT style commands. Any changes made will be retained even after power is removed from the EGBT-046S, hence device configuration setup must not be repeated unless new changes need to be made.

You can do configuration setup using the host controller itself (the microcontroller in your own circuit), or a PC running a terminal software using a serial to TTL (or USB to Serial TTL) converter. See Figure 4 for connection details.

It is important to note that EGBT-046S does not wait for any termination character for each AT command entry. Instead, it acts to whatever character you entered after one second. Hence, if you are not able to complete a command entry within a second, it will be ignored. Because of this behavior, it may be extremely difficult to do manual entry configuration using Windows Hyperterminal software. Terminal software that allows batch sending of multiple characters must be used.

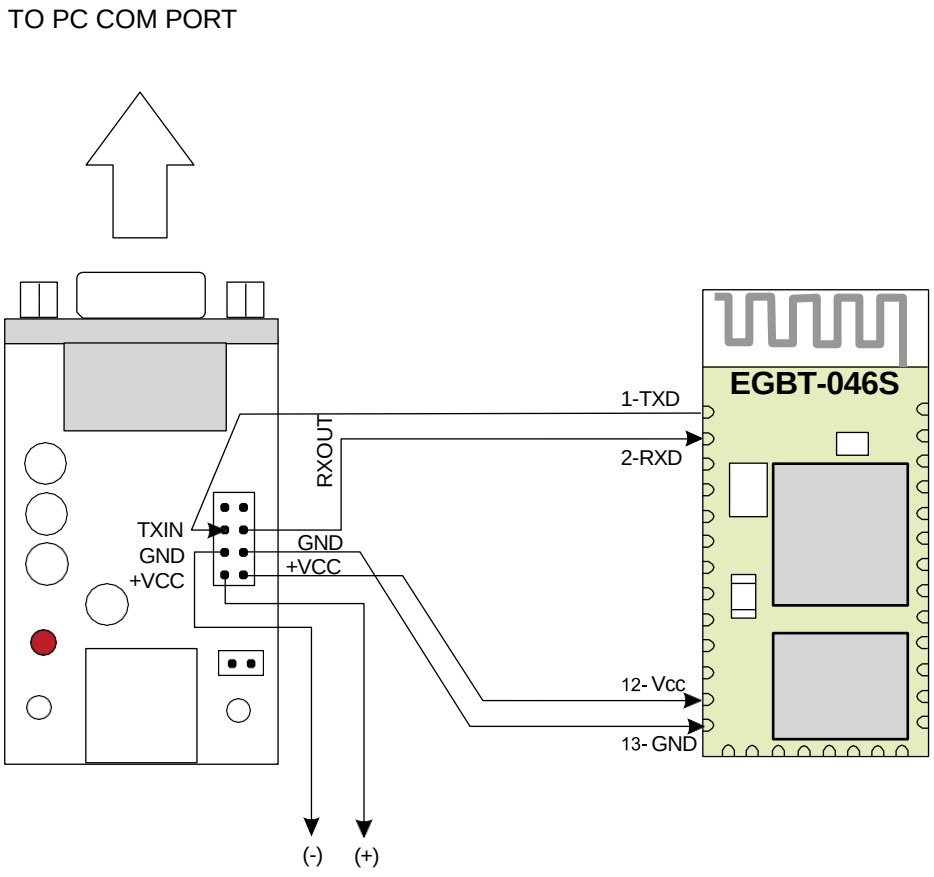


Figure 4. *A PC may be used to configure the EGBT-04. To connect to a PC COM port, a RS-232C to TTL converter is needed. This figure shows a wiring example using e-Gizmo RS-232 to TTL converter kit.*

EGBT-046S AT Command Set

1. TEST

Used to test the UART connection between the host controller and Bluetooth Module.

COMMAND	RESPONSE
AT	OK

2. Change Baud Rate

COMMAND	RESPONSE
AT+BAUD<p>	OK<r>

where:

<p> Paramater
<r> Response, set to nnnn bps

<p>	<r>	Remarks
1	1200	set to 1200bps
2	2400	set to 2400bps
3	4800	set to 4800bps
4	9600	set to 9600bps (Default)
5	19200	set to 19200bps
6	38400	set to 38400bps
7	57600	set to 57600bps
8	115200	set to 115200bps
9	230400	set to 230400bps
A	460800	set to 460800bps
B	921600	set to 921600bps
C	1382400	set to 1382400bps

Caution:
PC standard COM port hardware does not support baud rates in excess of 115200bps. If you are using a PC to configure EGBT-046S and accidentally set

baud rate to these values, connection to a PC COM port will no longer be possible. Use of USB to Serial converter cable that can work at higher bauds may be necessary to re-establish a connection. Prolific PL-2303 based USB to Serial converter cables are known to work up to 921600bps.

Example1: Set baud rate to 57600bps

From Host controller:
AT+BAUD7
EGBT-046S Response
OK57600

Example2: Set baud rate to 4800bps

From Host controller:
AT+BAUD3
EGBT-046S Response
OK4800

3. Change Device Name

The EGBT-046S can be assigned a readable name of up to 20 characters in lenght.

COMMAND	RESPONSE
AT+NAME<name>	OK<name>

Example1: Set device name as EGBT-04

From Host controller:
AT+NAMEEGBT-04
EGBT-046S Response
OKEGBT-04

4. Change PASSKEY(PIN code)

Passkey (PIN Code) is a 4-digit code shared with a master Bluetooth Device (e.g. PC) to prevent unauthorized pairing.

COMMAND	RESPONSE
AT+PIN<nnnn>	OK<nnnn>

Where:
<nnnn> 4-digit passkey

Example1: Set PASSKEY to 5995

From Host controller:
AT+PIN5995
EGBT-046S Response
OK5995