

UNIVERSIDAD NACIONAL DE CÓRDOBA
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES
CÁTEDRA DE ELECTRÓNICA DIGITAL III

TRABAJO PRÁCTICO INTEGRADOR

“SISTEMA DE RIEGO AUTOMÁTICO Y MONITOREO REAL”

Grupo Nº 11

Alumnos:
Grafion, Atilio Leonel
Oviedo, Ignacio Nicolás
Mendez, Jorge Nicolas

Profesor:
Migliore, Emiliano

Comisión Nº 3

TABLA DE CONTENIDO

- 1. PROYECTO**
- 2. DESCRIPCIÓN DEL SEP**
- 3. DESARROLLO DE HARDWARE**
 - 3.1. CIRCUITO DEL SSEP MICROCONTROLADOR**
 - 3.2. CIRCUITO DEL SSE COMUNICACIÓN DE DATOS**
 - 3.3. CIRCUITO DEL SSE ULTRASÓNICO HC-SR04**
 - 3.4. CIRCUITO DEL SSE CAPACITIVE MOISTURE SENSOR v1.2**
 - 3.5. CIRCUITO DEL SSE DE SEÑALIZACIÓN ÓPTICA Y SONORA**
 - 3.6. CIRCUITO DEL SSE DE MÓDULO MOSFET IRLZ44N**
- 4. DESARROLLO DE FIRMWARE**
 - 4.1. FIRMWARE DEL SSE CAPACITIVE MOISTURE SENSOR v1.2**
 - 4.2. FIRMWARE DEL SSE DE SEÑALIZACIÓN ÓPTICA Y SONORA**
 - 4.3. FIRMWARE DEL SSE DE BOMBA DE AGUA**
 - 4.4. FIRMWARE DEL SSR ULTRASONICO HC-SR04**
- 5. PRUEBAS DE SISTEMA**
- 6. CONCLUSIONES**
- 7. BIBLIOGRAFÍA Y REFERENCIAS**
- 8. ANEXO**
 - 8.1. INTERFAZ DE USUARIO**
 - 8.2. FIRMWARE DEL SEP**
 - 8.3. HOJAS DE DATOS**

TÉRMINOS Y DEFINICIONES

TPI: Trabajo Práctico Integrador

SEP: Sistema Electrónico Programable

SSE: Subsistema Electrónico

SSEP: Subsistema Electrónico Programable

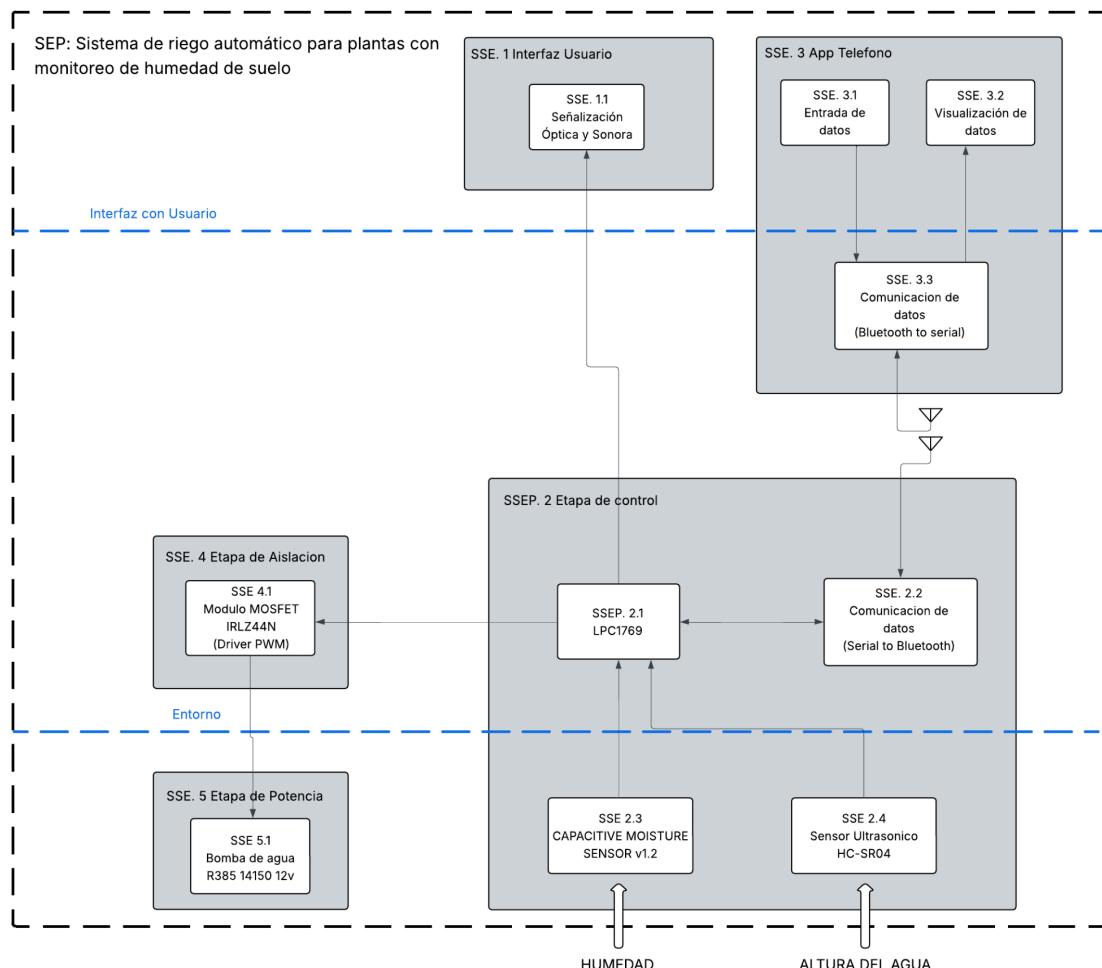
1. PROYECTO

El presente Trabajo Práctico Integrador describe la implementación de un **sistema de riego automático para plantas con monitoreo de humedad de suelo**.

El mismo se basa en un SSEP LPC1769 como sistema microcontrolador, el cual mide la humedad del suelo mediante un sensor capacitivo, y controla una bomba de agua a través de un módulo MOSFET con regulación de velocidad por PWM. Además, incorpora un sensor ultrasónico HC-SR04 para detectar el nivel de agua del tanque y evitar el funcionamiento de la bomba cuando el nivel es demasiado bajo.

El usuario puede supervisar el estado del sistema desde una aplicación móvil mediante Bluetooth, donde se visualiza tanto la humedad del suelo como el nivel del tanque. Desde la misma aplicación es posible configurar el límite de humedad, ajustar la velocidad de la bomba, y seleccionar entre dos modos de funcionamiento: manual (encendido y apagado directo de la bomba) y automático (controlado según la humedad medida).

Para la señalización local, el sistema utiliza un LED verde cuando el tanque tiene un nivel seguro de agua y un LED rojo acompañado de un buzzer cuando el nivel es inferior al 10%, bloqueando la bomba hasta que el tanque sea recargado.



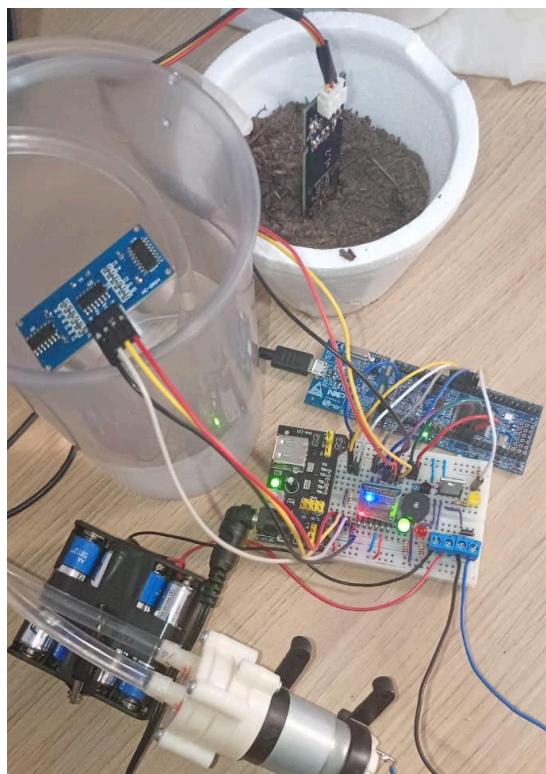
2. DESCRIPCIÓN DEL SEP

El sistema de monitoreo y control de la humedad del suelo funciona con una bomba de agua R385 14150 que se alimenta a 12V. Esta bomba es controlada por un módulo MOSFET de canal N (IRLZ44N), lo que permite ajustar su potencia a través de una señal PWM que proviene de la LPC1769. El sistema ofrece dos modos de operación que se pueden seleccionar desde una aplicación móvil, a través bluetooth: manual y automático. En el modo automático, las lecturas del sensor capacitivo de humedad del suelo v1.2 se comparan con un valor límite que el usuario ha configurado. Dependiendo de esta comparación, el sistema activa o detiene la bomba de forma autónoma para mantener la humedad deseada.

En el modo manual, el usuario tiene la opción de encender o apagar la bomba directamente desde la aplicación, sin que haya intervención automática, exceptuando cuando el límite de humedad es superado. En ambos modos, se puede ajustar la velocidad de la bomba mediante PWM, lo que permite un control más preciso sobre el flujo de riego.

Además de controlar la humedad, el sistema incluye un sensor ultrasónico HC-SR04 que mide el nivel de agua en el tanque de reserva. Este sensor actúa como una medida de protección: si el nivel de agua cae por debajo del umbral mínimo (equivalente al 10 % de la capacidad), el sistema apaga la bomba, la bloquea y activa una señal de advertencia local que incluye un LED rojo y un buzzer. Cuando el tanque se llena nuevamente por encima del umbral, se enciende un LED verde, la alarma se desactiva y la bomba se habilita para funcionar de nuevo.

Toda la información del sistema —porcentaje de humedad del suelo, nivel de agua del tanque y velocidad configurada de la bomba— se envía en tiempo real a la aplicación móvil, desde donde el usuario también puede ajustar los parámetros de funcionamiento y supervisar el estado general del sistema.



¿Cómo funciona el Sensor Capacitivo de Humedad del Suelo v1.2?

El sensor capacitivo de humedad del suelo funciona gracias a la variación en la permitividad dieléctrica del suelo, que cambia según la cantidad de agua que hay presente. Este dispositivo está compuesto por dos pistas conductoras que actúan como las placas de un condensador, mientras que el suelo actúa como el dieléctrico. Cuando la humedad aumenta, también lo hace la permitividad del medio, lo que incrementa la capacitancia entre las placas.

$$C = \epsilon \cdot \frac{A}{d}$$

C = Capacidad, ϵ = Permitividad dieléctrica (depende de la humedad),

A = Área efectiva, d = Distancia (entre las placas)

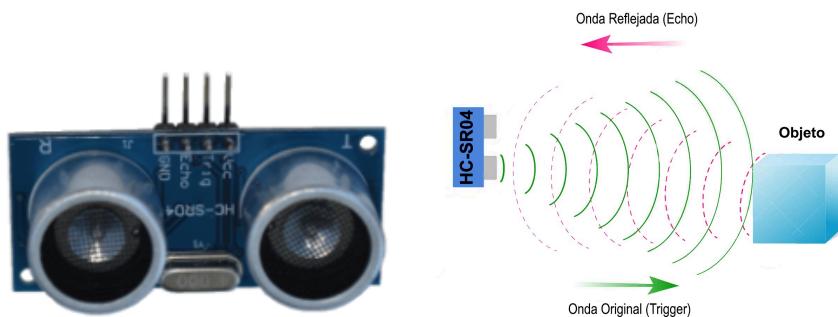
El circuito integrado en el sensor mide estos cambios de capacidad y convierte esa información en una señal analógica que es proporcional al nivel de humedad. Esta señal se puede leer a través del conversor analógico-digital (ADC) del microcontrolador. A diferencia de los sensores resistentes, el principio capacitivo evita la corrosión de los electrodos, lo que resulta en mediciones más estables y duraderas.



¿Cómo funciona el Sensor Ultrasónico HC SR04?

El sensor ultrasónico HC-SR04 funciona según el principio de propagación del sonido en un medio y la reflexión de las ondas acústicas. Este sensor emite ondas ultrasónicas, que son sonidos de alta frecuencia (mayores a 20 kHz, por lo que no son audibles para el oído humano) y viajan por el aire a una velocidad aproximada de 343 m/s a temperatura ambiente. Cuando estas ondas se encuentran con un obstáculo, se reflejan y regresan al emisor, donde el receptor las detecta. Al medir el tiempo que tarda la onda en ir y volver, el sistema puede calcular la distancia al objeto utilizando la ecuación del movimiento uniforme.

$$d = \frac{v \cdot t}{2}$$



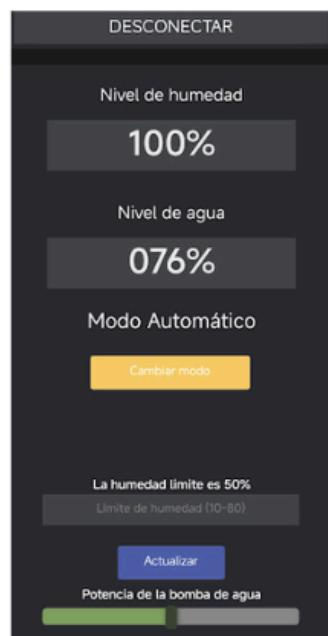
Interfaz de Usuario

La aplicación móvil permite conectarte al sistema a través de Bluetooth con solo presionar el botón “Conectar”, que cambia automáticamente a “Desconectar” una vez que la comunicación se establece correctamente. En la parte superior de la interfaz se encuentra el indicador “Nivel de humedad”, que muestra el porcentaje detectado por el sensor de humedad del suelo. Justo debajo, hay un segundo indicador similar, que muestra el nivel de agua en el tanque, también expresado en porcentaje.

Más abajo se ubica el indicador de modo, que informa si el sistema está funcionando en modo manual o automático, junto con el botón “Cambiar modo”, que permite alternar entre ambos. En el modo manual se activan los botones “Cargar” y “Detener”, además del indicador de estado de la bomba, que muestra si está encendida o apagada; en el modo automático, estos elementos se ocultan para evitar intervenciones manuales.

En la parte inferior de la interfaz hay un campo de texto donde se puede ingresar el valor límite de humedad, que por defecto es del 80 % y puede ajustarse entre 10 % y 80 %. Finalmente, por debajo de este campo se encuentra una barra deslizante que permite ajustar en tiempo real la potencia de la bomba de agua, sin importar el modo de operación.

Modo Automático

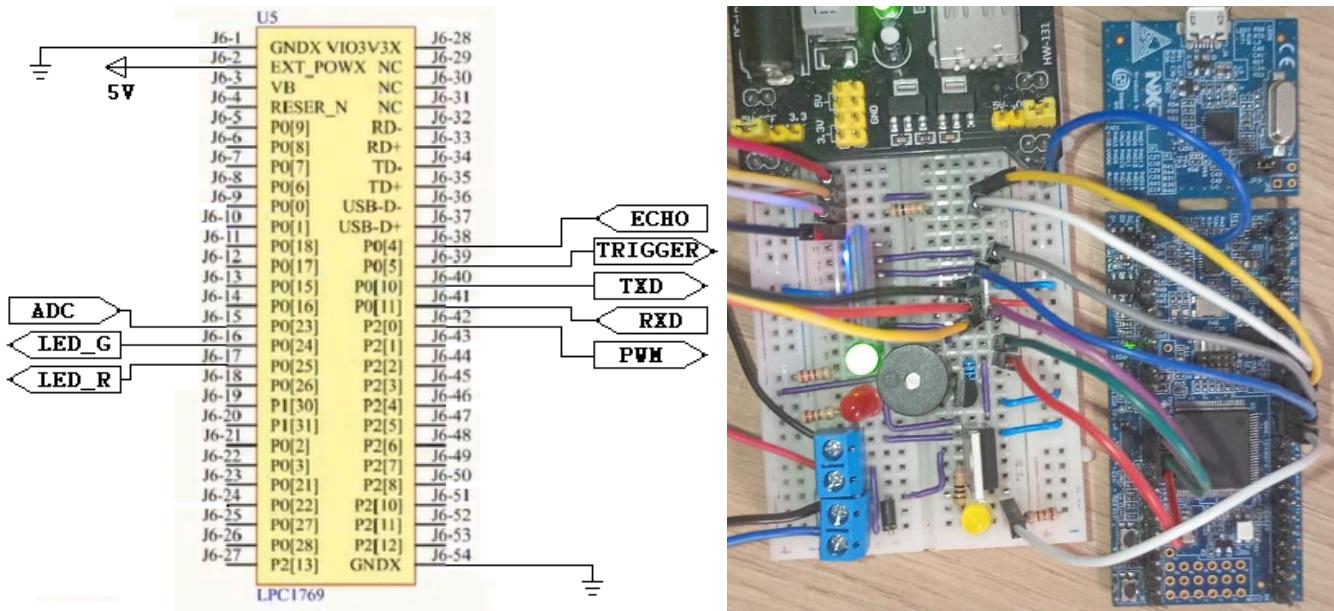


Modo Manual

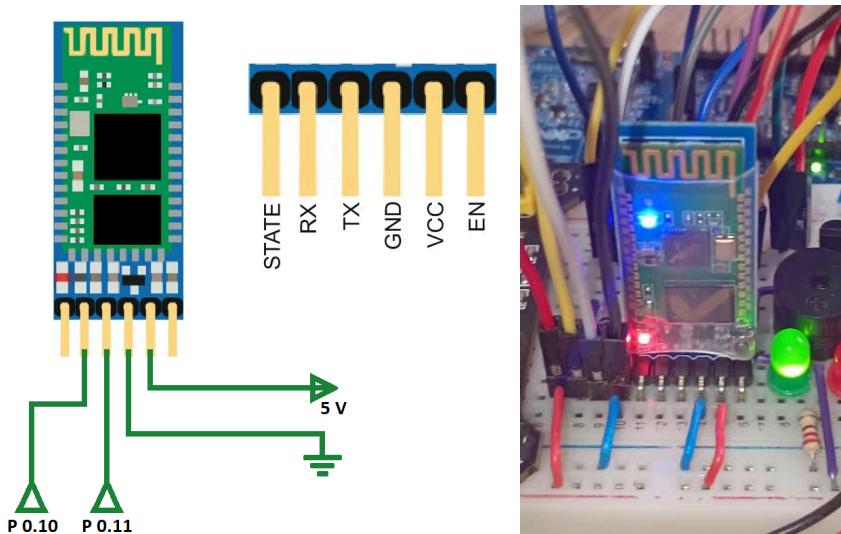


3. DESARROLLO DE HARDWARE

3.1. CIRCUITO DEL SSEP MICROCONTROLADOR



3.2. CIRCUITO DEL SSE COMUNICACIÓN DE DATOS



El módulo HC-05 es un módulo Bluetooth SPP (Protocolo de Puerto Serie) fácil de usar, diseñado para configurar conexiones serie inalámbricas.

Datos Básicos

- *Voltaje de operación:*
 - Núcleo: 1.8 V
 - I/O: 1.8 – 3.6 V
- *Frecuencia:* 2.4 GHz (banda ISM)
- *Sensibilidad:* -80 dBm
- *Potencia de transmisión:* +4 dBm máx
- *Interfaz:* UART
- *Baudrate por defecto:* 38400 bps, 8N1, sin paridad

- *Baudrates soportados:* 9600 – 460800 bps
- *Distancia:* hasta 10 m

PINOUT

- EN/KEY: Usado para modo AT
- VCC: 5V o 3.3V
- GND: GND
- TXD: RX de la LPC1769(P 0.11)
- RXD: TX de la LPC1769(P 0.10)
- STATE: Indicador de conexión (HIGH = enlazado, LOW=no enlazado)

Modos de operación

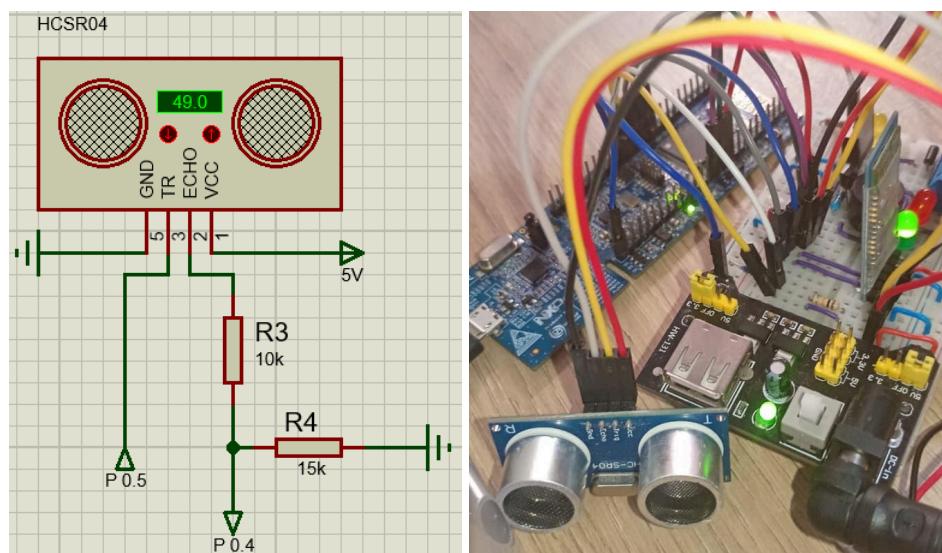
- Datos: Comunicación serie transparente
- Comunicación AT: Permite configurar el nombre, PIN, rol, velocidades.

Para entrar al modo AT, poner KEY en alto antes de alimentar el módulo.

Indicadores Led:

- Parpadeo rápido: sin conexión.
- Parpadeo lento: conectado.
- Parpadeo muy lento: modo AT.

3.3. CIRCUITO DEL SSE ULTRASÓNICO HC-SR04



El módulo HC-SR04 sirve para poder medir una distancia, mediante onda de sonido.

Datos Básicos

- Alimentación: 5V
- Corriente de funcionamiento: 10mA
- Distancia de medición: 2400 cm
- Resolución: 0,3 cm
- Ángulo de medición: 30°
- Dimensiones: 45 mm x 20 mm x 15 mm

- Peso: aprox. 10 g.

PINOUT

- VCC: 5V
- TRIG: P 0.5
- ECHO: P 0.4(con un divisor resistivo)
- GND: GND

Divisor Resistivo

Para proteger la entrada del microcontrolador (que opera a 3.3 V), es necesario reducir la señal de 5 V proveniente del sensor. Esto se realiza mediante un divisor resistivo. En este caso se utilizaron:

- R1 = 10 kΩ (conectada al pin de ECHO)
- R2 = 15 kΩ (conectada a GND)

La tensión de salida se obtiene con la fórmula del divisor resistivo:

$$V_{out} = V_{in} \cdot \frac{R_2}{R_1+R_2}$$

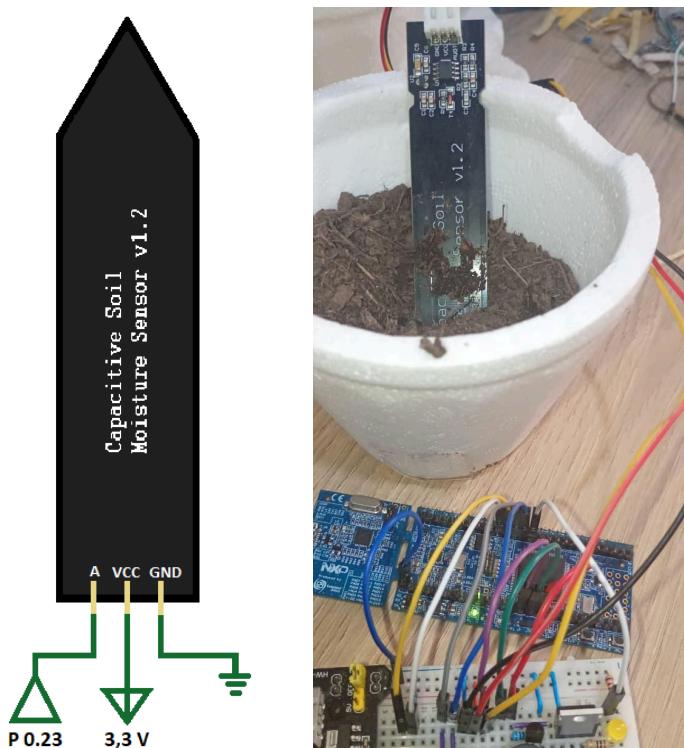
$$V_{out} = 5 V \cdot \frac{15k\Omega}{10k\Omega+15k\Omega}$$

$$V_{out} = 5 V \cdot \frac{15}{25}$$

$$V_{out} = 5 V \cdot 0,6 = 3V$$

La señal original queda reducida a 3 V, que es segura y compatible con entradas lógicas de 3.3 V del microcontrolador.

3.4. CIRCUITO DEL SSE CAPACITIVE MOISTURE SENSOR v1.2



Datos Básicos

- Sensor capacitivo de humedad de suelo (no se corroa).
- Alimentación: 3.3 – 5.5V.
- Salida analógica: 0 – 3.0V (inversamente proporcional a la humedad).
- Corriente de funcionamiento: 5mA
- Tamaño: 98 × 23 mm.
- Peso: aprox. 30 g.

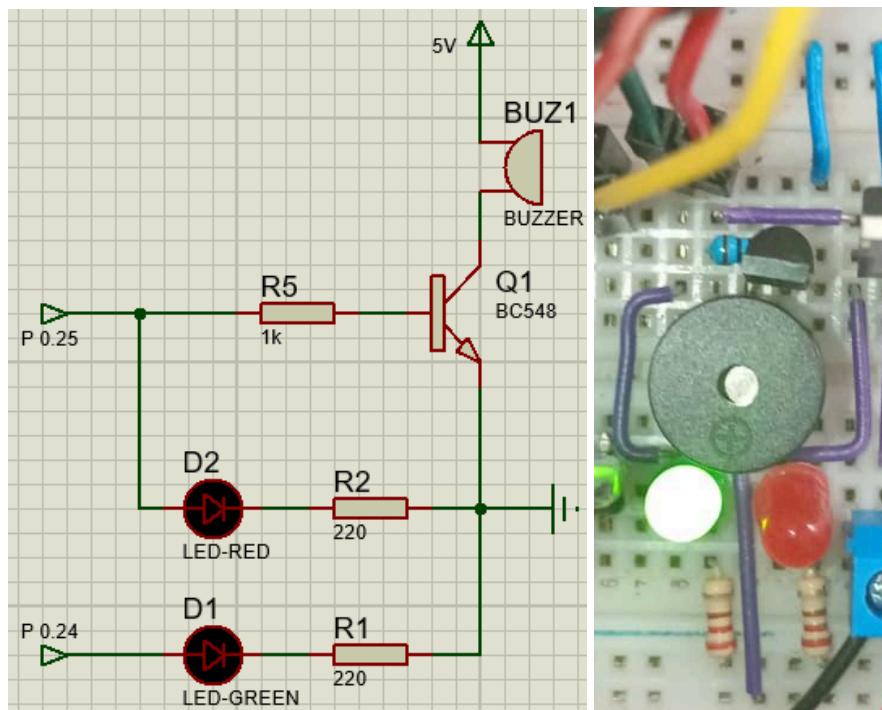
PINOUT

- VCC: 3,3V - 5V
- GND: GND
- AOUT: P 0.23 (ADC 0.0)

Funcionamiento

- Más seco → mayor valor analógico.
- Más húmedo → menor valor analógico.

3.5. CIRCUITO DEL SSE DE SEÑALIZACIÓN ÓPTICA y SONORA



La señalización óptica se realiza mediante dos LEDs: uno verde, que se enciende cuando el nivel del tanque supera el 10 % de su capacidad, y uno rojo, que se activa cuando el nivel cae por debajo de ese valor. Además, cuando el nivel del agua está por debajo del 10 %, se habilita también la señalización sonora a través de un buzzer, que alerta al usuario sobre la falta de agua en el tanque.

Cálculo de resistencia de Base

$$I_{CZ} = 30mA \text{ (según datasheet Buzzer)}$$

$$I_c = 100mA \text{ y } I_b = 5mA \text{ (sat Transistor)}$$

$$\frac{I_c}{I_b} = \frac{100mA}{5mA} = 20 \text{ (relación de sat)}$$

$$I_{bz} = \frac{I_{cz}}{20} = \frac{30mA}{20} = 1,5mA$$

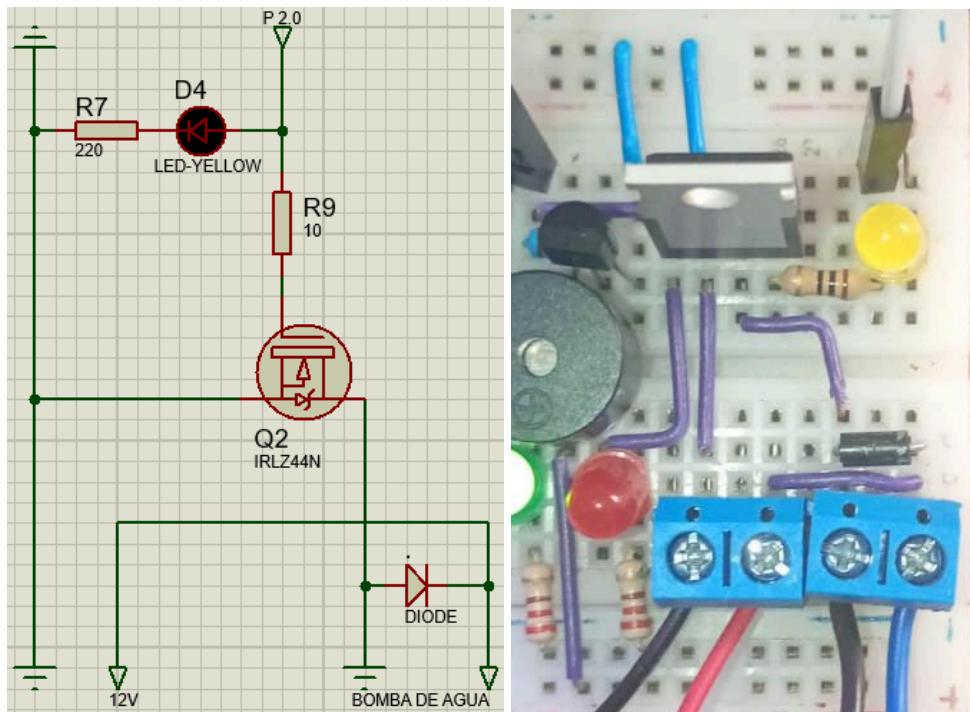
$$V_{gpio} = 3,3V$$

$$V_{be} = 0,7V$$

$$R_b = \frac{V_{gpio}-V_{be}}{I_{bz}} = \frac{3,3V-0,7}{1,5mA} = 1,73k\Omega$$

Por lo tanto usamos una resistencia de $1k\Omega$, la cual permite saturación, aunque aumenta un poco más la corriente base, sin afectar a la LPC1769.

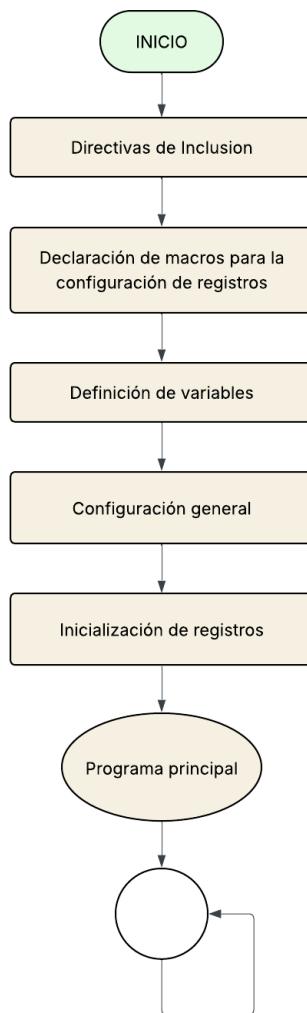
3.6. CIRCUITO DEL SSE DE MÓDULO MOSFET IRLZ44N



El módulo MOSFET permite adaptar la señal PWM de 0–3,3 V generada por la LPC1769 para controlar la bomba de agua de 12V. Para ello se utiliza un MOSFET IRLZ44N, un transistor de canal N apto para lógica de 3,3 V. En su entrada de compuerta se coloca una resistencia de 10Ω , cuyo propósito es limitar el pico de corriente que aparece cuando la gate del transistor se carga en cada conmutación del PWM. Sin esta resistencia, el GPIO del microcontrolador estaría expuesto a corrientes instantáneas muy elevadas.

El módulo también tiene un diodo flyback conectado en paralelo con la bomba, cuya función es proteger al MOSFET de los picos de corriente generados por la carga inductiva al apagar la bomba. El módulo también incluye un LED indicador amarillo, que señala el estado de activación del MOSFET.

4. DESARROLLO DE FIRMWARE



4.1. FIRMWARE DEL SSE CAPACITIVE MOISTURE SENSOR v1.2

La lógica de funcionamiento del sensor Capacitive Moisture v1.2 para la captura y control de la humedad en el suelo utiliza el canal 0 del ADC y el timer 1 en modo match. La configuración del Timer 1 genera cada 1 segundos un toggle en la salida del pin relacionado al match channel 0. Se configura el ADC para iniciar una conversión con el evento MAT1.0 y por flanco descendente, de este modo, se realiza una conversión cada 2 segundos. En la lógica de interrupción del ADC se guarda la muestra, se cargan los datos en el buffer de transferencia, se actualiza el estado del sistema y se reinicia el canal 1 de DMA para transferir la nueva medición de humedad hacia la aplicación. Dentro de la actualización del sistema, si el sistema se encuentra en suspenso (pause = 1 porque el nivel de agua es menor al umbral de seguridad) se apaga la bomba de agua y si el sistema está en modo automático se realiza un chequeo de la humedad medida y se enciende o apaga la bomba de agua.

Para evitar que la bomba se encienda y apague rápidamente cuando la humedad está cerca del límite se aplica una histéresis() que varía según el valor de humedad límite configurado. Es mínima cuando se establece un límite bajo (menor a 10%) y varía progresivamente hasta un máximo cuando se configura un valor alto (80%).

$$Histéresis = 5 + \frac{H-10}{70} * 10$$

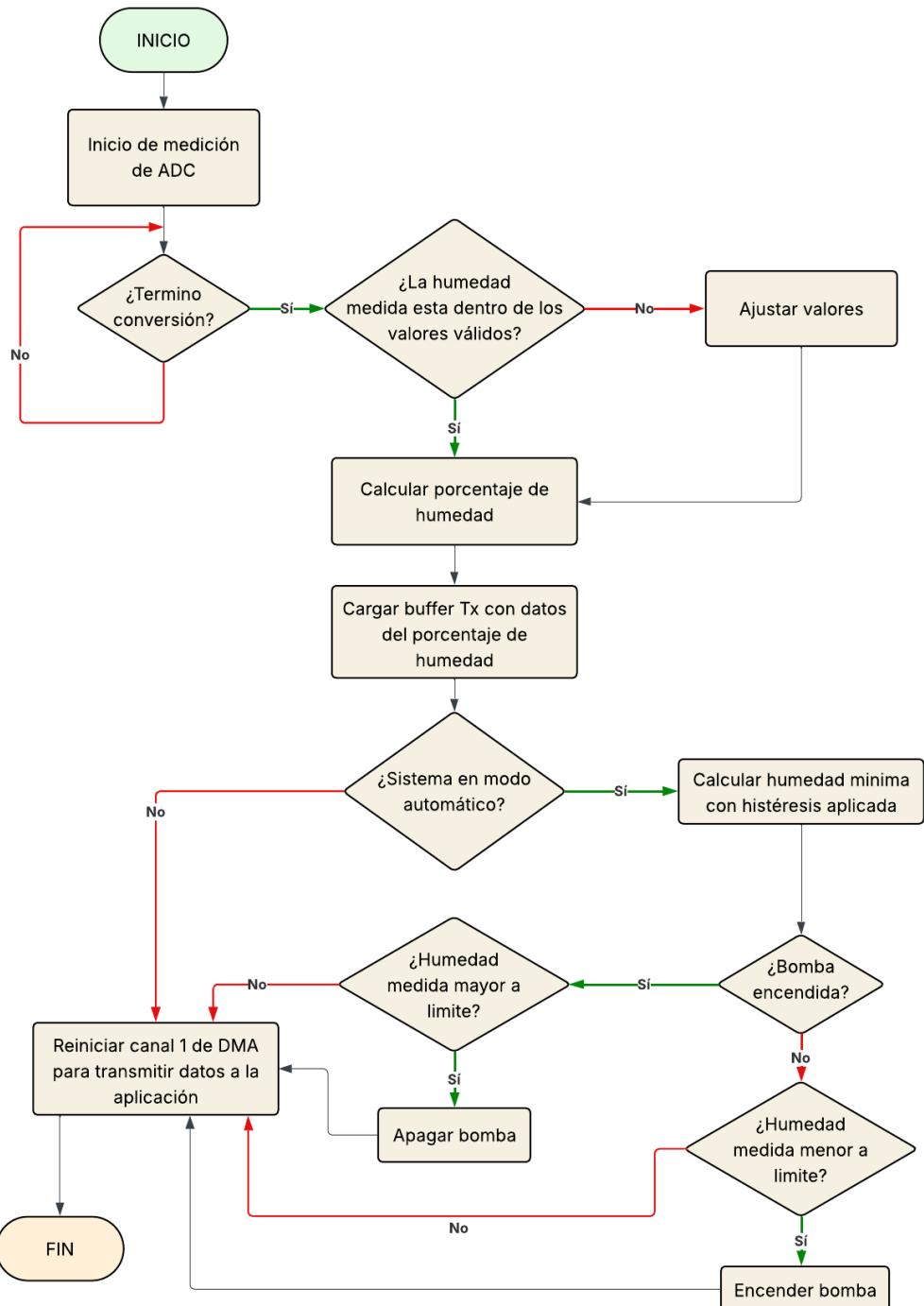
5 : Histéresis mínima

15: Histéresis máxima

10-80: Rango de humedad válido

H: valor de humedad límite

Histéresis: Es una técnica para evitar que el sistema encienda y apague constantemente cuando una señal está cerca de un valor límite. Consiste en establecer dos umbrales distintos, uno para encender y uno para apagar.



Rutina de interrupción del ADC

```

void cfgADC(void)
{
    ADC_Init(LPC_ADC,100000);
    ADC_BurstCmd(LPC_ADC, DISABLE);
    ADC_StartCmd(LPC_ADC, ADC_START_ON_MAT10);
    ADC_ChannelCmd(LPC_ADC, ADC_CHANNEL_0, ENABLE);

    ADC_EdgeStartConfig(LPC_ADC, ADC_START_ON_FALLING);
    ADC_IntConfig(LPC_ADC, ADC_ADINTEN0, SET);

    NVIC_SetPriority(ADC_IRQn, 5);
    NVIC_EnableIRQ(ADC_IRQn);
}

void ADC_IRQHandler(void)
{
    while(!(ADC_ChannelGetStatus(LPC_ADC, ADC_CHANNEL_0, ADC_DATA_DONE))){}

    adc_sample = ADC_ChannelGetData(LPC_ADC, ADC_CHANNEL_0) & 0xFFFF;

    setMoisturePercent();
    systemUpdate();

    GPDMA_ChannelCmd(1, DISABLE);
    cfgDmaUart();
    GPDMA_ChannelCmd(1, ENABLE);
}

void setMoisturePercent(void)
{
    if(adc_sample >= MAX_VAL_MOIS) moisPrct = 0;
    else if(adc_sample <= MIN_VAL_MOIS) moisPrct = 100;
    else moisPrct = ((uint32_t)(MAX_VAL_MOIS - adc_sample) * 100) / RANGE_VAL_MOIS;

    TxBuffer[0] = convertNumHex(moisPrct / 100);      // centena
    TxBuffer[1] = convertNumHex((moisPrct / 10) % 10); // decena
    TxBuffer[2] = convertNumHex(moisPrct % 10);        // unidad
}

void systemUpdate(void)
{
    if(pause){
        pumpOff();
        return;
    }

    if(autoFlg)checkMoisture();
}

void checkMoisture(void)
{
    uint8_t minMois = (percent - getHysteresis());

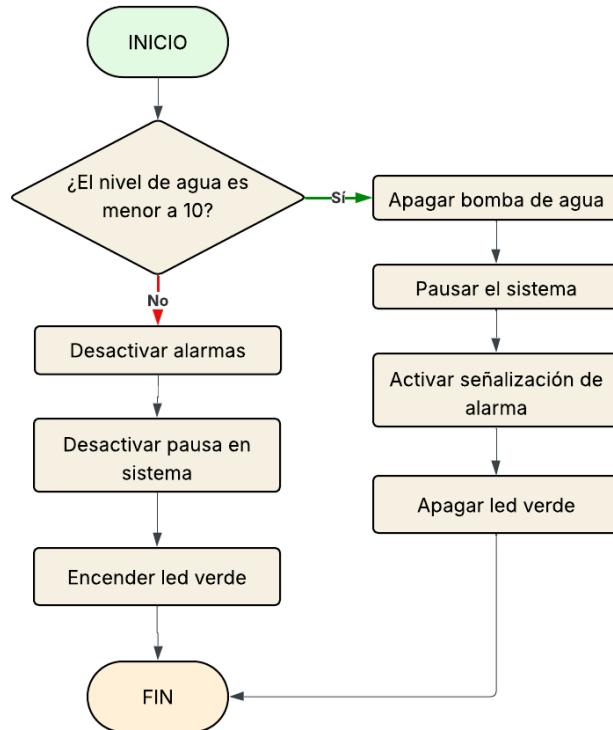
    if (pump_on)
    {
        if (moisPrct >= percent)pumpOff(); //detener cuando alcanzamos o superamos la humedad deseada
    }
    else
    {
        if (moisPrct <= minMois)pumpOn(); //arrancar solo cuando la humedad cae por debajo del limite inferior
    }
}

uint8_t getHysteresis()
{
    return (uint8_t)(5 + ((uint32_t)(percent - 10) * 10) / 70);
}

```

4.2. FIRMWARE DEL SSE DE SEÑALIZACIÓN ÓPTICA Y SONORA

La lógica de activación del sistema de señalización se desarrolla dentro del control del nivel de agua, activando señalización de alarma cuando el nivel de agua en el depósito es menor al umbral de seguridad y desactivándolo cuando el nivel está dentro de un valor válido. Cuando se activa la señalización de alarma (salida RED LED), se apaga la salida del led verde (GREEN_LED) y viceversa.



Lógica de función `lowWaterLevel`

```

void lowWaterLevel(void)
{
    if(wtr_lvl < 10)
    {//pause
        pumpOff();
        pause = 1;
        //alarm
        alarmOn();
    }
    else
    {//resume
        alarmOff();
        pause = 0;
    }
}

void alarmOn(void)
{
    GPIO_ClearValue(PORT_0, GREEN_LED);

    TIM_Cmd(LPC_TIM0, ENABLE);
}

void alarmOff(void)
{
    TIM_ResetCounter(LPC_TIM0);
    TIM_Cmd(LPC_TIM0, DISABLE);
    GPIO_ClearValue(PORT_0, RED_LED);

    GPIO_SetValue(PORT_0, GREEN_LED);
}

```

4.3. FIRMWARE DEL SSE DE BOMBA DE AGUA

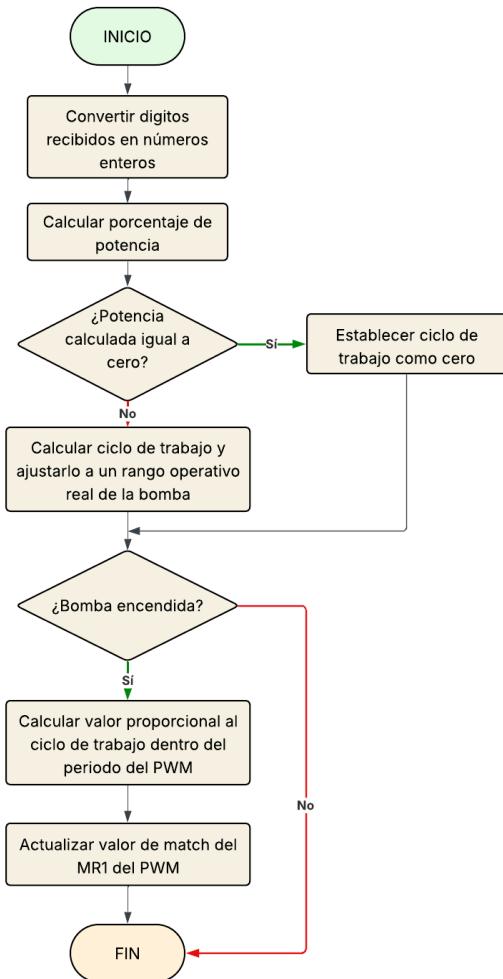
La generación de la señal PWM se realiza utilizando el periférico PWM1. Se configura el temporizador interno del PWM eligiendo el prescaler PWM_TIMER_PRESCALE_TICKVAL y estableciendo un PrescaleValue = 1. De esta manera, el contador TC se incrementa a la misma frecuencia que el reloj del periférico PWM (PCLK_PWM1 = CCLK / 4), lo que permite obtener la máxima resolución temporal.

El registro **MR0** se configura para determinar el **período del PWM**. Se usa la constante PWM_PERIOD = 25000 para que el temporizador se reinicie cuando el contador TC alcance este valor. Como el PWM está funcionando con PCLK_PWM1, este valor establece un período correspondiente a una frecuencia de **1 kHz**. Se configura el registro **MR1**, que define el **ciclo de trabajo (duty cycle)**. Este registro provoca un cambio de estado en la salida PWM cuando el contador TC coincide con su valor, determinando así cuánto tiempo la señal permanece en nivel alto dentro de cada período.

La función setDuty se utiliza para modificar el ciclo de trabajo, así se puede controlar la velocidad de la bomba ajustando el tiempo en el que la señal PWM permanece en alto dentro de cada período. Esta función recibe un porcentaje de ciclo de trabajo, calcula un valor proporcional dentro del período del PWM y lo usa para modificar el valor de match del MR1 del PWM en ese instante.

Para establecer la potencia de la bomba de agua se usa la función setPotency. Esta función convierte los dígitos recibidos por UART y almacenados en el buffer RxBuffer

en números enteros. Se aplica una transformación para usar un rango operativo real de la bomba porque esta no puede funcionar en ciclos de trabajos muy bajos, se utiliza un ciclo mínimo de 35%.



Lógica de función `setPotency`

```

void cfgPWM(void)
{
    PWM_TIMERCFG_Type cfgPWMTimer;
    PWM_MATCHCFG_Type cfgPWM_MR0; // Mat 0 - PERIODO
    PWM_MATCHCFG_Type cfgPWM_MR1; // Mat 1 - DUTY

    cfgPWMTimer.PrescaleOption = PWM_TIMER_PRESCALE_TICKVAL;
    cfgPWMTimer.PrescaleValue = 1;
    PWM_Init(LPC_PWM1, PWM_MODE_TIMER, &cfgPWMTimer);

    //Configuración de MR0 (Periodo)
    cfgPWM_MR0.MatchChannel = 0;
    cfgPWM_MR0.IntOnMatch = DISABLE;
    cfgPWM_MR0.StopOnMatch = DISABLE;
    cfgPWM_MR0.ResetOnMatch = ENABLE; // MR0 REINICIA el contador
    PWM_ConfigMatch(LPC_PWM1, &cfgPWM_MR0);
    PWM_MatchUpdate(LPC_PWM1, 0, PWM_PERIOD, PWM_MATCH_UPDATE_NOW);

    //Configuración de MR1 (Duty)
    cfgPWM_MR1.MatchChannel = 1;
    cfgPWM_MR1.IntOnMatch = DISABLE;
    cfgPWM_MR1.StopOnMatch = DISABLE;
    cfgPWM_MR1.ResetOnMatch = DISABLE; // MR1 NO debe reiniciar el contador
    PWM_ConfigMatch(LPC_PWM1, &cfgPWM_MR1);
    PWM_MatchUpdate(LPC_PWM1, 1, 0, PWM_MATCH_UPDATE_NOW);

    PWM_CounterCmd(LPC_PWM1, ENABLE);
    PWM_ChannelCmd(LPC_PWM1, 1, ENABLE);
    PWM_Cmd(LPC_PWM1, ENABLE);
}

void setDuty(uint8_t duty_percent)
{
    if (duty_percent > 100) duty_percent = 100;

    uint32_t new_match_value = (PWM_PERIOD * duty_percent) / 100;
    PWM_MatchUpdate(LPC_PWM1, 1, new_match_value, PWM_MATCH_UPDATE_NOW);
}

void pumpOn(void)
{
    pump_on = 1;
    setDuty(duty);
}

void pumpOff(void)
{
    pump_on = 0;
    setDuty(0);
}

void setPotency(void)
{
    uint8_t pot = convertHexNum(RxBuffer[1]) * 100 +
                  convertHexNum(RxBuffer[2]) * 10 +
                  convertHexNum(RxBuffer[3]);

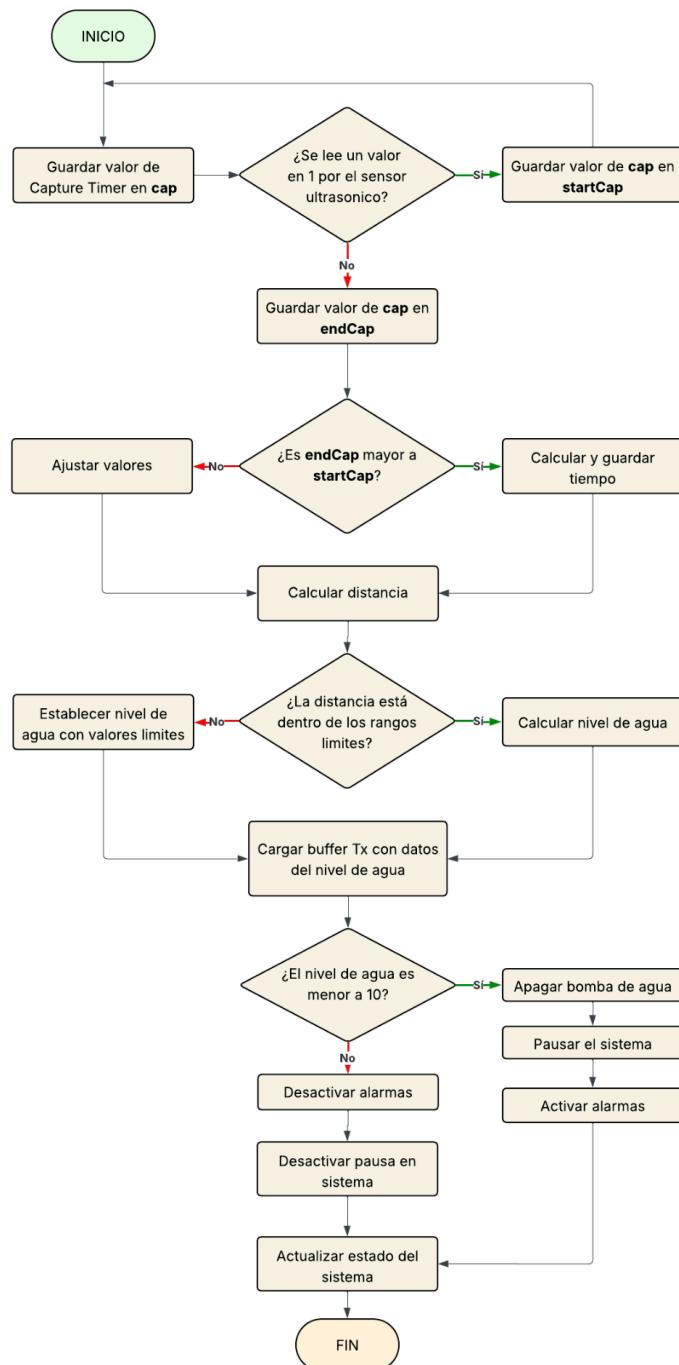
    if(pot == 0)duty = 0;
    else duty = 35 + (65 * pot) / 100;

    if(pump_on)setDuty(duty);
}

```

4.4. FIRMWARE DEL SSE ULTRASÓNICO HC-SR04

La lógica de funcionamiento del sensor HC-SR04 para el control del nivel de agua utiliza el módulo Systick y el Timer 2 en modo capture. En la lógica de interrupción del Systick (configurado para realizar una interrupción cada 10us) se utiliza una variable estática para enviar una señal de salida por el **TRIGGER** del sensor (P0.5) con un periodo de 100 ms. El sensor obtiene el rebote de la señal emitida por el trigger por medio del **ECHO** conectado al pin (P0.4) dedicado al Timer 2 en modo capture. Se configuró el modo capture para generar una interrupción por flanco ascendente y descendente, de este modo se procesa cuando el pin ECHO del sensor emita y deje de emitir señal. La lógica de interrupción del timer 2 calcula el nivel de agua del recipiente, carga el buffer de datos para la transmisión hacia la aplicación, realiza el control del nivel de agua medido para encender o apagar la señalización de alarma y actualiza el estado del sistema.



```

void SysTick_Handler(void)//cada 10 us y periodo de 100ms
{
    static uint32_t vTiks = 0;

    if(vTiks < 1) GPIO_SetValue(PORT_0, TRIG);
    else GPIO_ClearValue(PORT_0, TRIG);

    vTiks = (vTiks + 1) % 10000;

    SysTick -> CTRL &= SysTick -> CTRL;
}

void TIMER2_IRQHandler(void)
{
    uint32_t cap = TIM_GetCaptureValue(LPC_TIM2, 0);

    if(GPIO_ReadValue(PORT_0) & ECHO)
    {
        startCap = cap;
    }
    else
    {
        endCap = cap;

        if(endCap >= startCap)distanceTime = endCap - startCap;
        else distanceTime = (0xFFFFFFFF - startCap) + endCap + 1; //si se pasa

        distance = (distanceTime * 343) / 20000;

        setWaterLevel();
        lowWaterLevel();
        systemUpdate();
    }

    TIM_ClearIntCapturePending(LPC_TIM2, TIM_CRO_INT);
}

void setWaterLevel(void)
{
    if(distance >= 30) wtr_lvl = 0;
    else if(distance <= 5) wtr_lvl = 100;
    else wtr_lvl = ((uint32_t)(30 - distance) * 100) / (30 - 5);

    TxBuffer[5] = convertNumHex(wtr_lvl / 100);      // centena
    TxBuffer[6] = convertNumHex((wtr_lvl / 10) % 10); // decena
    TxBuffer[7] = convertNumHex(wtr_lvl % 10);        // unidad
}

void lowWaterLevel(void)
{
    if(wtr_lvl < 10)
    {///pause
        pumpOff();
        pause = 1;
        ///alarma
        alarmOn();
    }
    else
    {///resume
        alarmOff();
        pause = 0;
    }
}

```

5. PRUEBAS DE SISTEMA

Nro. Caso de Prueba	Descripción del caso de prueba	Paso	Resultado Esperado	Resultado obtenido	Observaciones
1	Activación del sistema de riego estando en modo automático y utilizando el porcentaje de humedad predeterminado.	Encender el SSEP y cargar el depósito de agua por encima del umbral de seguridad.	La bomba de agua no debería encender para mediciones de humedad mayores a 80%	Pasa	
		Colocar el sensor en un suelo cuya humedad sea menor al 80%	La bomba de agua debería encenderse al medir un porcentaje de agua menor al 80%.	Pasa	
2	Activación de señalización visual de nivel de agua por debajo del umbral de seguridad.	Encender el SSEP y cargar el depósito de agua por encima del umbral de seguridad.	Cambiar a modo manual y encender la bomba.	Pasa	
		Mantener la bomba de agua encendida.	La bomba de agua debería apagarse cuando la medición del nivel de agua sea menor al umbral de seguridad y encender la alarma y led rojo.	Pasa	
3	Activación de señalización visual de nivel de agua por encima del umbral de seguridad.	Encender el SSEP y cargar el depósito de agua por encima del umbral de seguridad.	El led verde debe encenderse y mantenerse encendido mientras el nivel del agua sea mayor al umbral de seguridad.	Pasa	
4	Activación del sistema de riego en modo manual	Encender el SSEP y cargar el depósito de agua por encima del umbral de seguridad.	El estado de modo en la aplicación está en modo Automático	Pasa	

		Presionar el botón 'Cambiar modo'	El estado de modo actual en la aplicación cambia a Manual	Pasa	
		Presionar el botón 'Recargar'	La bomba de agua debería encender.	Pasa	
5	Detener el funcionamiento de la bomba de agua en modo manual	Encender el SSEP, cargar el depósito de agua por encima del umbral de seguridad y cambiar a modo Manual.	El estado de modo actual en la aplicación cambia a Manual y la bomba debería estar apagada	Pasa	
		Presionar el botón 'Recargar'	La bomba de agua debería encender.	Pasa	
		Presionar el botón 'Detener'	La bomba de agua debería detenerse.	Pasa	
6	La bomba de agua no enciende cuando el nivel de agua del depósito es menor al umbral de seguridad en modo manual.	Encender el SSEP, dejar el nivel de agua del depósito por debajo del umbral de seguridad y cambiar a modo Manual.	El estado de modo en la aplicación cambia a modo Manual y la bomba está detenida.	Pasa	
		Presionar el botón 'Recargar'	La bomba de agua no enciende mientras el nivel de agua del depósito esté debajo del umbral de seguridad.	Pasa	
7	Aumentar potencia de bomba a 90% en modo Manual.	Encender el SSEP y cargar el depósito de agua por encima del umbral de seguridad.	La bomba de agua no debería encender para mediciones de humedad mayores a 50%	Pasa	
		Cambiar a modo Manual	El estado de modo en la aplicación cambia a modo Manual y la bomba está detenida.	Pasa	

		Presionar el botón 'Recargar'	La bomba de agua debería encenderse	Pasa	
		Mover la barra de potencia de bomba en la aplicación hasta un valor de 90%	La potencia de la bomba de agua debería aumentar	Pasa	
8	Actualizar el porcentaje límite de humedad en modo Automatico	Encender el SSEP y cargar el depósito de agua por encima del umbral de seguridad.	La bomba de agua no debería encender para mediciones de humedad mayores a 80%	Pasa	
		Ingresar el valor 20 como nuevo límite de humedad y presionar el botón 'Actualizar'	El valor de humedad límite se actualiza. La bomba no debería encender para mediciones de humedad mayores a 20%.	Pasa	
9	El nivel de humedad se muestra en la aplicación	Encender el SSEP y cargar el depósito de agua por encima del umbral de seguridad.	En la aplicación debería mostrarse el porcentaje de humedad actual	Pasa	
9	El nivel de agua se muestra en la aplicación	Encender el SSEP y cargar el depósito de agua por encima del umbral de seguridad.	En la aplicación debería mostrarse el porcentaje de nivel de agua en el depósito.	Pasa	

6. CONCLUSIONES

El desarrollo de este proyecto nos permitió entender de manera práctica cómo se pueden integrar sensores y control digital en un sistema real. Durante el proceso, nos encontramos con varios desafíos que nos llevaron a analizar, corregir y mejorar el diseño, lo que resultó en una experiencia valiosa sobre cómo se comportan los componentes y la importancia de proteger tanto la placa como la carga. Además, la interacción con la aplicación móvil nos hizo reflexionar sobre la usabilidad del sistema, no sólo en términos de su funcionamiento técnico. En resumen, el TPI nos enseñó que un sistema confiable depende tanto de un buen diseño electrónico como de un manejo adecuado de la lógica de control y de las condiciones reales de operación. También abrió la puerta a futuras mejoras y ampliaciones, demostrando que siempre hay espacio para optimizar y perfeccionar el proyecto.

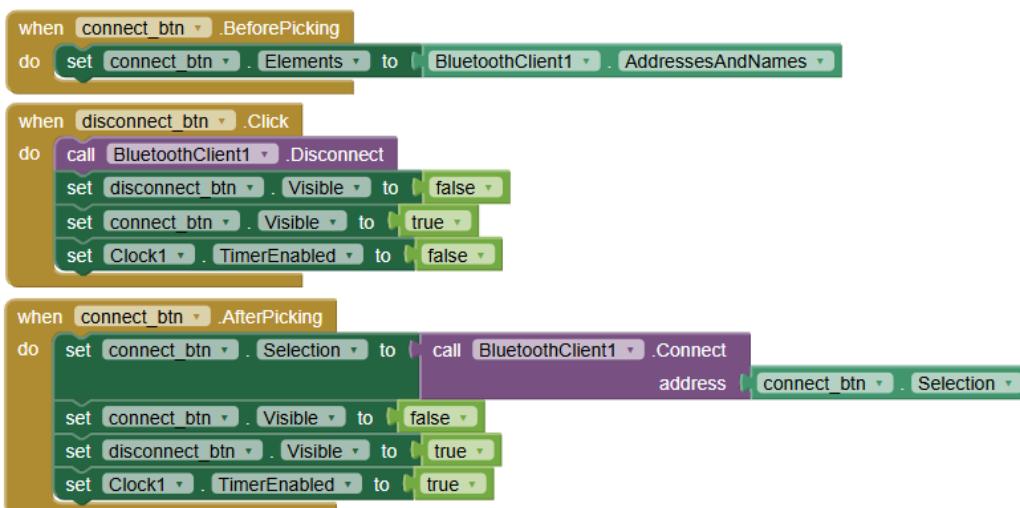
7. BIBLIOGRAFÍA Y REFERENCIAS

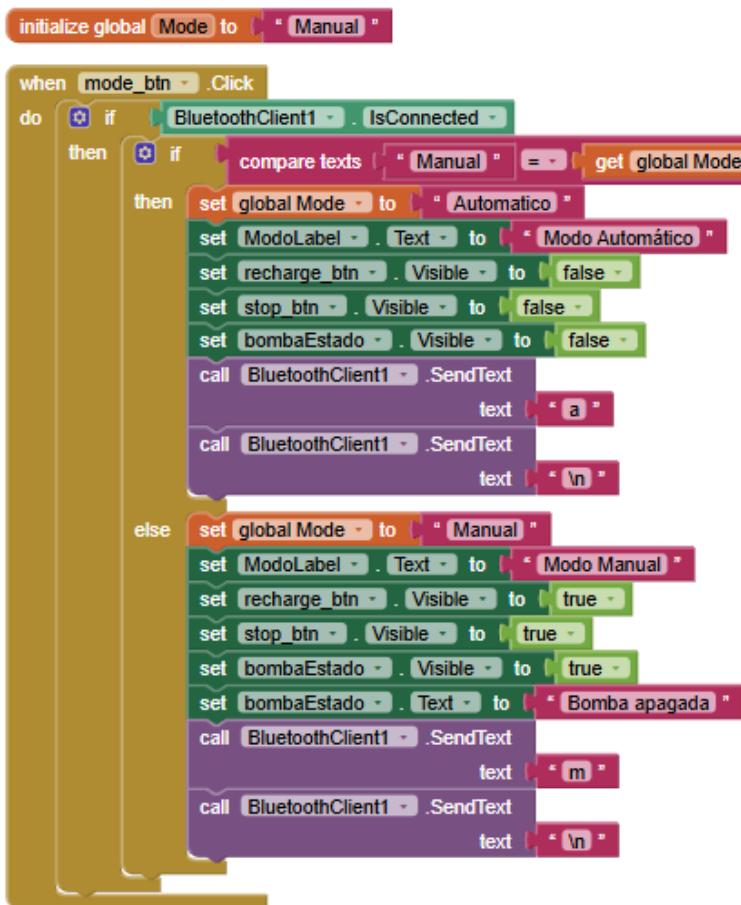
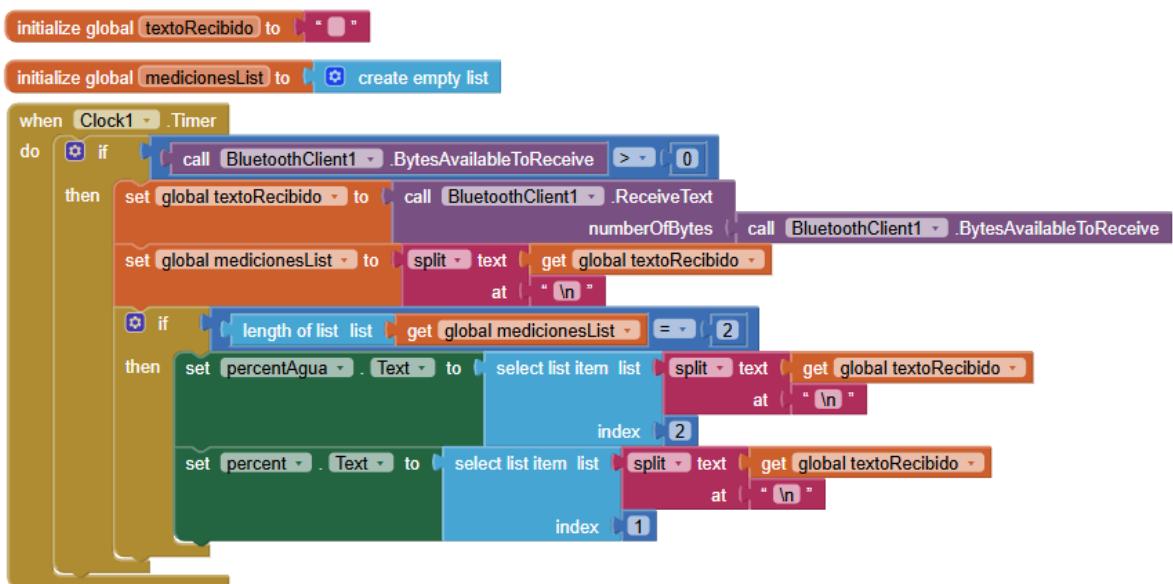
- [1] Microchip Technology Inc. (2010). *PIC16F882/883/884/886/887 Data Sheet* (DS41291D). Obtenido de <https://ww1.microchip.com/downloads/en/devicedoc/41291d.pdf>
- <https://www.canva.com/design/DAG0jyJe4rQ/0tsNZGuCInr6shkzr1zrhg/edit>
- <https://www.canva.com/design/DAGzjlKpXfo/j38RYMt5VIxep-4RcKsuAg/edit>
- <https://www.lorric.com/es/Articles/flowmeter-technology/flowmeter-technology/hysteresis-function>

8. ANEXO

8.1. INTERFAZ DE USUARIO

En el desarrollo de la aplicación móvil se utiliza la herramienta online MIT App Inventor que provee una forma sencilla, eficaz e intuitiva de crear una aplicación móvil y programar su lógica mediante bloques de código. Con esta herramienta se diseñó la interfaz gráfica de la aplicación y se implementó la lógica para realizar el cambio de Modo, mostrar el porcentaje de humedad y enviar datos hacia el módulo UART mediante una conexión Bluetooth.





```

when recharge_btn .Click
do
  if BluetoothClient1 . IsConnected
    then set bombaEstado . Text to "Bomba encendida"
        call BluetoothClient1 . SendText
          text "e"
        call BluetoothClient1 . SendText
          text "\n"

```

```

when stop_btn .Click
do
  if BluetoothClient1 . IsConnected
    then set bombaEstado . Text to "Bomba apagada"
        call BluetoothClient1 . SendText
          text "d"
        call BluetoothClient1 . SendText
          text "\n"

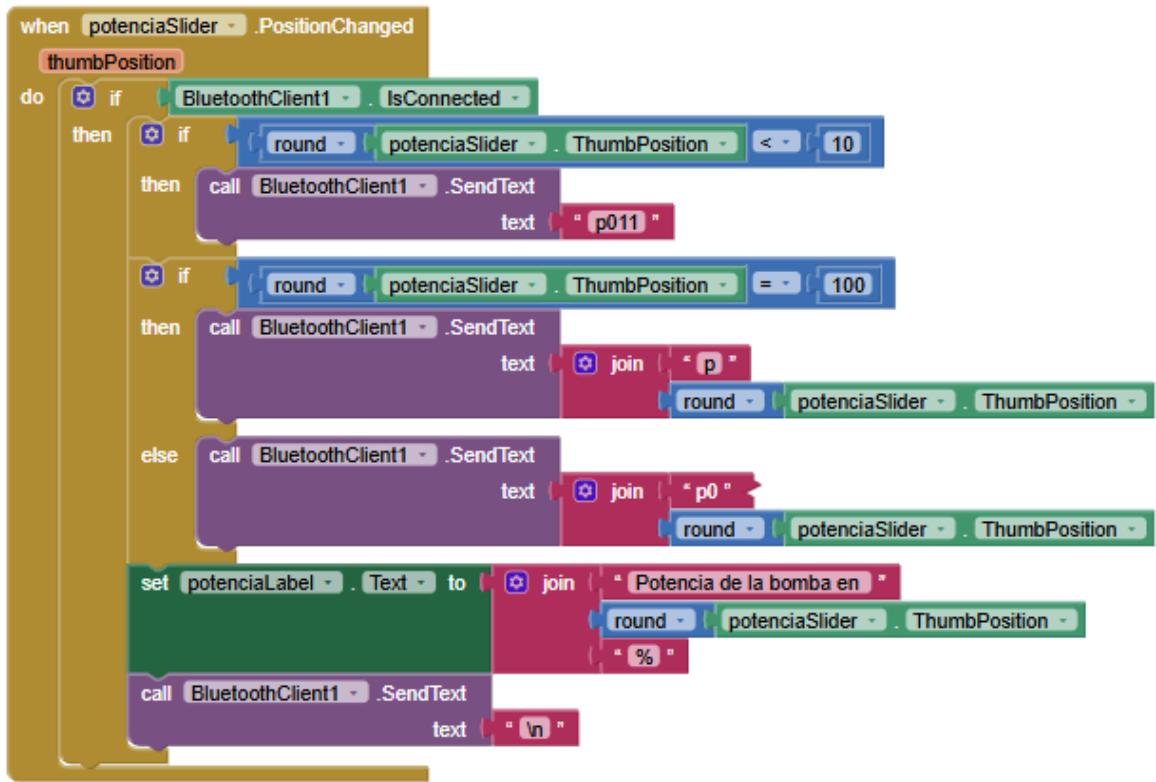
```

```
initialize global Humedad to 50
```

```

when update_btn .Click
do
  if BluetoothClient1 . IsConnected
    then
      if is empty humedadValor . Text
        then set error_label . Text to "¡Ingrese un valor para la humedad!"
      else
        if is number? humedadValor . Text
          then
            if humedadValor . Text ≥ 10 and humedadValor . Text ≤ 80
              then set global Humedad to humedadValor . Text
                  set humedadLimite . Text to join " La humedad límite es de "
                    get global Humedad
                    "%"
                  call BluetoothClient1 . SendText
                    text "h0"
                  call BluetoothClient1 . SendText
                    text get global Humedad
                  set error_label . Text to ""
                  call BluetoothClient1 . SendText
                    text "\n"
            else set error_label . Text to "El valor debe estar entre 10% - 80%"
            else set error_label . Text to "El valor de la humedad debe ser un número!"

```



8.2. FIRMWARE DEL SEP

Código LPC1769

```

/*
* Programa para control de riego automatico controlado por bluetooth con
LPC1769
* Configuracion:
* - Bomba de agua de 12V controlado por PWM: P2.0
* - Sensor de humedad de suelo : P0.23
* - Sensor de distancia HC-SR04 : P0.4(ECHO pin) - P0.5(TRIG pin)
* - Modulo Bluetooth HC-05 : P0.10(RX sensor) - P0.11(TX sensor)
* - LED Rojo/Buzzer : P0.25
* - LED Verde : P0.24
*/
#include "LPC17xx.h"
#include "../Drivers/inc/lpc17xx_pinsel.h"
#include "../Drivers/inc/lpc17xx_gpio.h"
#include "../Drivers/inc/lpc17xx_uart.h"
#include "../Drivers/inc/lpc17xx_timer.h"
#include "../Drivers/inc/lpc17xx_gpdma.h"
#include "../Drivers/inc/lpc17xx_adc.h"
#include "../Drivers/inc/lpc17xx_pwm.h"
//=====
//===== MACROS Y VARIABLES =====
//=====

```

```

//Macros
#define OUTPUT      (uint8_t)1
#define INPUT       (uint8_t)0
#define PORT_0      (uint8_t)0
#define PORT_2      (uint8_t)2
#define PUMP        (uint32_t)(1 << 0)
#define RED_LED     (uint32_t)(1 << 25)
#define GREEN_LED   (uint32_t)(1 << 24)
#define TRIG        (uint32_t)(1 << 5)
#define ECHO        (uint32_t)(1 << 4)
#define TX_BUFFER_SIZE 10
#define RX_BUFFER_SIZE 4
#define PWM_PERIOD 25000
//-----Variables-----
//ADC
volatile uint16_t adc_sample = 0;
//UART
uint8_t TxBuffer[TX_BUFFER_SIZE] = {0, 0, 0, 0x25, 0x0A, 0, 0, 0, 0x25,
0x0A}; //0x25 es % y 0x0A es salto de linea
uint8_t RxBuffer[RX_BUFFER_SIZE] = {0, 0, 0, 0};
volatile uint8_t receivedData = 0; //dato recibido
//sensor de distancia
volatile uint32_t distanceTime = 0;
volatile uint32_t distance = 0;
volatile uint32_t startCap = 0;
volatile uint32_t endCap = 0;
uint8_t wtr_lvl = 0;
//control de la bomba
uint8_t autoFlg = 0;
uint8_t pump_on = 0;
uint8_t pause = 0;
uint8_t duty = 55; //duty cicle inicial (PWM)
//sensor de humedad de suelo
const uint16_t MAX_VAL_MOIS = 3474;
const uint16_t MIN_VAL_MOIS = 2052;
const uint16_t RANGE_VAL_MOIS = MAX_VAL_MOIS - MIN_VAL_MOIS;
uint8_t moisPrct = 0; //porcentaje actual
uint8_t percent = 80; //porcentaje maximo
uint8_t minMois = 10;
//=====
//=====CONFIGURACIONES=====
//=====
//-----
//Configuracion de los pines
//-----
void cfgPin(void){
    //PWM1.1
    PINSEL_CFG_Type cfgPumpPin = {PINSEL_PORT_2, PINSEL_PIN_0,
PINSEL_FUNC_1, PINSEL_PINMODE_PULLDOWN, PINSEL_PINMODE_NORMAL};
    PINSEL_ConfigPin(&cfgPumpPin);
}

```

```

    //TRIG PIN
    PINSEL_CfgType cfgTrigPin = {PINSEL_PORT_0, PINSEL_PIN_5,
PINSEL_FUNC_0, PINSEL_PINMODE_PULLDOWN, PINSEL_PINMODE_NORMAL};
    PINSEL_ConfigPin(&cfgTrigPin);
    GPIO_SetDir(0, 1 << 5, 1);
    GPIO_ClearValue(0, 1 << 5);
    //TIMER2 - CAP2.0/ECHO
    PINSEL_CfgType cfgTimerCapPin = {PINSEL_PORT_0, PINSEL_PIN_4,
PINSEL_FUNC_3, PINSEL_PINMODE_PULLDOWN, PINSEL_PINMODE_NORMAL}; //P0.4
    PINSEL_ConfigPin(&cfgTimerCapPin);
    //UART
    PINSEL_CfgType cfgTXD2Pin    = {PINSEL_PORT_0, PINSEL_PIN_10,
PINSEL_FUNC_1, 0, PINSEL_PINMODE_NORMAL};
    PINSEL_ConfigPin(&cfgTXD2Pin);
    PINSEL_CfgType cfgRXD2Pin    = {PINSEL_PORT_0, PINSEL_PIN_11,
PINSEL_FUNC_1, 0, PINSEL_PINMODE_NORMAL};
    PINSEL_ConfigPin(&cfgRXD2Pin);
    //TIMER0 - MAT0.0
    PINSEL_CfgType cfgMat0Pin_CH0 = {PINSEL_PORT_1, PINSEL_PIN_28,
PINSEL_FUNC_3, 0,PINSEL_PINMODE_NORMAL};
    PINSEL_ConfigPin(&cfgMat0Pin_CH0);
    //TIMER1 - MAT1.0
    PINSEL_CfgType cfgMat1Pin_CH0 = {PINSEL_PORT_1, PINSEL_PIN_22,
PINSEL_FUNC_3, 0,PINSEL_PINMODE_NORMAL};
    PINSEL_ConfigPin(&cfgMat1Pin_CH0);
    //ADC - AD0.0
    PINSEL_CfgType cfgAD0Pin_CH0 = {PINSEL_PORT_0, PINSEL_PIN_23,
PINSEL_FUNC_1, PINSEL_PINMODE_TRISTATE, PINSEL_PINMODE_NORMAL};
    PINSEL_ConfigPin(&cfgAD0Pin_CH0);
    //GREEN LED
    PINSEL_CfgType cfgGreenLedPin = {PINSEL_PORT_0, PINSEL_PIN_24,
PINSEL_FUNC_0, PINSEL_PINMODE_PULLDOWN, PINSEL_PINMODE_NORMAL};
    PINSEL_ConfigPin(&cfgGreenLedPin);
    GPIO_SetDir(PORT_0, GREEN_LED, OUTPUT);
    GPIO_ClearValue(PORT_0, GREEN_LED);
    //RED LED
    PINSEL_CfgType cfgRedLedPin = {PINSEL_PORT_0, PINSEL_PIN_25,
PINSEL_FUNC_0, PINSEL_PINMODE_PULLDOWN, PINSEL_PINMODE_NORMAL};
    PINSEL_ConfigPin(&cfgRedLedPin);
    GPIO_SetDir(PORT_0, RED_LED, OUTPUT);
    GPIO_ClearValue(PORT_0, RED_LED);
}
//-----
//Configuracion de Timers
//-----
void cfgTimer0(void){
    //configuracion modo del timer
    TIM_TIMERCFG_Type cfgTimer0Mode;
    cfgTimer0Mode.PrescaleOption = TIM_PRESCALE_USVAL;
    cfgTimer0Mode.PrescaleValue = 1000;
}

```

```

//configuracion de MATCH0.0
TIM_MATCHCFG_Type cfgTimerMatch00;
cfgTimerMatch00.MatchChannel      = 0;
cfgTimerMatch00.MatchValue       = 999; //RECORDA
CAMBIAR POR 4999
cfgTimerMatch00.IntOnMatch      = ENABLE;
cfgTimerMatch00.ResetOnMatch    = ENABLE;
cfgTimerMatch00.StopOnMatch     = DISABLE;
cfgTimerMatch00.ExtMatchOutputType = TIM_EXTMATCH_NOTHING;
//inicializacion del timer
TIM_Init(LPC_TIM0, TIM_TIMER_MODE, &cfgTimer0Mode);
TIM_ConfigMatch(LPC_TIM0, &cfgTimerMatch00);
TIM_Cmd(LPC_TIM0, DISABLE);
//configuracion NVIC
NVIC_SetPriority(TIMER0_IRQn, 1);
}

void cfgTimer1(void){
//configuracion modo del timer
TIM_TIMERCFG_Type cfgTimer1Mode;
cfgTimer1Mode.PrescaleOption = TIM_PRESCALE_USVAL;
cfgTimer1Mode.PrescaleValue = 1000;
//configuracion de MATCH0.0
TIM_MATCHCFG_Type cfgTimerMatch10;
cfgTimerMatch10.MatchChannel      = 0;
cfgTimerMatch10.MatchValue       = 999;
cfgTimerMatch10.IntOnMatch      = DISABLE;
cfgTimerMatch10.ResetOnMatch    = ENABLE;
cfgTimerMatch10.StopOnMatch     = DISABLE;
cfgTimerMatch10.ExtMatchOutputType = TIM_EXTMATCH_TOGGLE;
//inicializacion del timer
TIM_Init(LPC_TIM1, TIM_TIMER_MODE, &cfgTimer1Mode);
TIM_ConfigMatch(LPC_TIM1, &cfgTimerMatch10);
TIM_Cmd(LPC_TIM1, ENABLE);
}

void cfgTimer2(void)
{
    TIM_TIMERCFG_Type cfgTimer2Mode;
    cfgTimer2Mode.PrescaleOption = TIM_PRESCALE_USVAL;
    cfgTimer2Mode.PrescaleValue = 1;
    TIM_CAPTURECFG_Type cfgTimerCap2;
    cfgTimerCap2.CaptureChannel = 0;
    cfgTimerCap2.RisingEdge = ENABLE;
    cfgTimerCap2.FallingEdge = ENABLE;
    cfgTimerCap2.IntOnCaption = ENABLE;
    TIM_Init(LPC_TIM2, TIM_TIMER_MODE, &cfgTimer2Mode);
    TIM_ConfigCapture(LPC_TIM2, &cfgTimerCap2);
    TIM_Cmd(LPC_TIM2, ENABLE);
    NVIC_SetPriority(TIMER2_IRQn, 2);
    NVIC_EnableIRQ(TIMER2_IRQn);
}

```

```

//-----
//Configuracion de PWM
//-----
void cfgPWM(void)
{
    PWM_TIMERCFG_Type cfgPWMTimer;
    PWM_MATCHCFG_Type cfgPWM_MR0; // Mat 0 - PERIODO
    PWM_MATCHCFG_Type cfgPWM_MR1; // Mat 1 - DUTY
    cfgPWMTimer.PrescaleOption = PWM_TIMER_PRESCALE_TICKVAL;
    cfgPWMTimer.PrescaleValue = 1;
    PWM_Init(LPC_PWM1, PWM_MODE_TIMER, &cfgPWMTimer);
    //Configuración de MR0 (Periodo)
    cfgPWM_MR0.MatchChannel = 0;
    cfgPWM_MR0.IntOnMatch = DISABLE;
    cfgPWM_MR0.StopOnMatch = DISABLE;
    cfgPWM_MR0.ResetOnMatch = ENABLE; // MR0 REINICIA el contador
    PWM_ConfigMatch(LPC_PWM1, &cfgPWM_MR0);
    PWM_MatchUpdate(LPC_PWM1, 0, PWM_PERIOD, PWM_MATCH_UPDATE_NOW);
    //Configuración de MR1 (Duty)
    cfgPWM_MR1.MatchChannel = 1;
    cfgPWM_MR1.IntOnMatch = DISABLE;
    cfgPWM_MR1.StopOnMatch = DISABLE;
    cfgPWM_MR1.ResetOnMatch = DISABLE; // MR1 NO debe reiniciar el
    //contador
    PWM_ConfigMatch(LPC_PWM1, &cfgPWM_MR1);
    PWM_MatchUpdate(LPC_PWM1, 1, 0, PWM_MATCH_UPDATE_NOW);
    PWM_CounterCmd(LPC_PWM1, ENABLE);
    PWM_ChannelCmd(LPC_PWM1, 1, ENABLE);
    PWM_Cmd(LPC_PWM1, ENABLE);
}
//-----
//Configuracion del UART
//-----
void cfgUart(void)
{
    //configuracion inicial de UART
    UART_CFG_Type cfgUART2;
    UART_ConfigStructInit(&cfgUART2); //Inicializa con 9600 de
    //baudrate, 8 bits de datos, sin bit de paridad
    UART_Init(LPC_UART2, &cfgUART2);
    //configuracion de FIFO
    UART_FIFO_CFG_Type cfgUART2FIFO;
    UART_FIFOConfigStructInit(&cfgUART2FIFO);
    cfgUART2FIFO.FIFO_DMAMode = ENABLE;
    UART_FIFOConfig(LPC_UART2, &cfgUART2FIFO);
    //configuracion de interrupcion por UART
    UART_IntConfig(LPC_UART2, UART_INTCFG_RBR, ENABLE); //interrupcion
    //por recepcion
    UART_TxCmd(LPC_UART2, ENABLE);
    //configuracion NVIC
}

```

```

        NVIC_SetPriority(UART2 IRQn, 4);
        NVIC_EnableIRQ(UART2 IRQn);
    }
//-----
//Configuracion del ADC
//-----
void cfgADC(void)
{
    ADC_Init(LPC_ADC,100000);
    ADC_BurstCmd(LPC_ADC, DISABLE);
    ADC_StartCmd(LPC_ADC, ADC_START_ON_MAT10);
    ADC_ChannelCmd(LPC_ADC, ADC_CHANNEL_0, ENABLE);
    ADC_EdgeStartConfig(LPC_ADC, ADC_START_ON_FALLING);
    ADC_IntConfig(LPC_ADC, ADC_ADINTEN0, SET);
    NVIC_SetPriority(ADC IRQn, 5);
    NVIC_EnableIRQ(ADC IRQn);
}
//-----
//Configuracion del DMA
//-----
//M2P
void cfgDmaUart(void)
{
    NVIC_DisableIRQ(DMA IRQn);
    GPDMA_Init();
    GPDMA_Channel_CFG_Type cfgDmaUART;
    cfgDmaUART.ChannelNum = 1;
    cfgDmaUART.TransferSize = TX_BUFFER_SIZE;
    cfgDmaUART.TransferWidth = 0;
    cfgDmaUART.SrcMemAddr = (uint32_t)TxBuffer;
    cfgDmaUART.DstMemAddr = 0;
    cfgDmaUART.TransferType = GPDMA_TRANSFERTYPE_M2P;
    cfgDmaUART.SrcConn = 0;
    cfgDmaUART.DstConn = GPDMA_CONN_UART2_Tx;
    cfgDmaUART.DMALLI = 0;
    GPDMA_Setup(&cfgDmaUART);
}
//=====
//=====FUNCIONES=====
//=====
void setDuty(uint8_t duty_percent)
{
    if (duty_percent > 100) duty_percent = 100;
    uint32_t new_match_value = (PWM_PERIOD * duty_percent) / 100;
    PWM_MatchUpdate(LPC_PWM1, 1, new_match_value, PWM_MATCH_UPDATE_NOW);
}
uint8_t getHysteresis()
{
    return (uint8_t)(5 + ((uint32_t)(percent - 10) * 10) / 70);
}

```

```

void pumpOn(void)
{
    pump_on = 1;
    setDuty(duty);
}
void pumpOff(void)
{
    pump_on = 0;
    setDuty(0);
}
void alarmOn(void)
{
    GPIO_ClearValue(PORT_0, GREEN_LED);
    TIM_Cmd(LPC_TIM0, ENABLE);
    NVIC_EnableIRQ(TIMER0 IRQn);
}
void alarmOff(void)
{
    TIM_ResetCounter(LPC_TIM0);
    TIM_Cmd(LPC_TIM0, DISABLE);
    NVIC_DisableIRQ(TIMER0 IRQn);
    GPIO_ClearValue(PORT_0, RED_LED);
    GPIO_SetValue(PORT_0, GREEN_LED);
}
void checkMoisture(void)
{
    minMois = (percent - getHysteresis());
    if (pump_on)
    {
        if (moisPrct >= percent)pumpOff(); //detener cuando alcanzamos o
superamos la humedad deseada
    }
    else
    {
        if (moisPrct <= minMois)pumpOn(); //arrancar solo cuando la
humedad cae por debajo del límite inferior
    }
}
void systemUpdate(void)
{
    if(pause){
        pumpOff();
        return;
    }
    if(autoFlg)checkMoisture();
    else if (moisPrct >= percent)pumpOff();
}
void selectMode(uint8_t modeSel)
{
    switch(modeSel)

```

```

    {
        case 'a'://modo automatico
            autoFlg = 1;
            break;
        case 'm'://modo manual
            if(!autoFlg)break;
            pumpOff();
            autoFlg = 0;
            break;
    }
}

void manualControl(uint8_t pmCtrl)
{
    switch(pmCtrl)
    {
        case 'd'://off
            pumpOff();
            break;
        case 'e'://on
            checkMoisture();
            break;
    }
}

void lowWaterLevel(void)
{
    if(wtr_lvl < 10)
    {//pause
        pumpOff();
        pause = 1;
        //alarma
        alarmOn();
    }
    else
    {//resume
        alarmOff();
        pause = 0;
    }
}

uint8_t convertNumHex(uint8_t conv)
{
    return 0x30 + conv;
}

uint8_t convertHexNum(uint8_t conv)
{
    return conv - 0x30;
}

/*
* maximo y minimo deseados (10% - 80%)
* 3332 -> 10%
* 2336 -> 80%

```

```

/*
void setMoisturePercent(void)
{
    if(adc_sample >= MAX_VAL_MOIS) moisPrct = 0;
    else if(adc_sample <= MIN_VAL_MOIS) moisPrct = 100;
    else moisPrct = ((uint32_t)(MAX_VAL_MOIS - adc_sample) * 100) /
RANGE_VAL_MOIS;
    TxBuffer[0] = convertNumHex(moisPrct / 100);           // centena
    TxBuffer[1] = convertNumHex((moisPrct / 10) % 10); // decena
    TxBuffer[2] = convertNumHex(moisPrct % 10);           // unidad
}
void setWaterLevel(void)
{
    if(distance >= 30) wtr_lvl = 0;
    else if(distance <= 5) wtr_lvl = 100;
    else wtr_lvl = ((uint32_t)(30 - distance) * 100) / (30 - 5);
    TxBuffer[5] = convertNumHex(wtr_lvl / 100);           // centena
    TxBuffer[6] = convertNumHex((wtr_lvl / 10) % 10); // decena
    TxBuffer[7] = convertNumHex(wtr_lvl % 10);           // unidad
}
void setPotency(void)
{
    uint8_t pot = convertHexNum(RxBuffer[1]) * 100 +
                  convertHexNum(RxBuffer[2]) * 10 +
                  convertHexNum(RxBuffer[3]);
    if(pot == 0)duty = 0;
    else duty = 35 + (65 * pot) / 100;
    if(pump_on)setDuty(duty);
}
void setMaxMoisture(void)
{
    percent = convertHexNum(RxBuffer[1]) * 100 +
              convertHexNum(RxBuffer[2]) * 10 +
              convertHexNum(RxBuffer[3]);
    if(percent > 80)percent = 80;
    if(percent < 10)percent = 0;
}
//=====
//=====HANDLERS=====
//=====

void UART2_IRQHandler(void)
{
    if(UART_GetLineStatus(LPC_UART2) & UART_LSR_RDR){
        if(pause) return;
        receivedData = UART_ReceiveByte(LPC_UART2);
        static uint8_t rxIdx = 0;
        uint8_t finishRx = 0;
        //Fin por ENTER
        if(receivedData == '\n' || receivedData == '\r')finishRx
= 1;
}

```

```

        else
        {
            //Limita de 4 Bytes max
            if (rxIdx < 4)RxBuffer[rxIdx++] = receivedData;
            if (rxIdx == 4)finishRx = 1;
        }
        if (finishRx)
        {
            if (rxIdx >= 1 && rxIdx <= 4)
            {
                if(RxBuffer[0] == 'h')setMaxMoisture();
                if(RxBuffer[0] == 'p')setPotency();
                systemUpdate();
                selectMode(RxBuffer[0]);
                if(!autoFlg)manualControl(RxBuffer[0]);
            }
            rxIdx = 0;
        }
    }
}

void ADC_IRQHandler(void)
{
    while(!(ADC_ChannelGetStatus(LPC_ADC, ADC_CHANNEL_0,
ADC_DATA_DONE))){}
    adc_sample = ADC_ChannelGetData(LPC_ADC, ADC_CHANNEL_0) & 0xFFFF;
    setMoisturePercent();
    systemUpdate();
    GPDMA_ChannelCmd(1, DISABLE);
    cfgDmaUart();
    GPDMA_ChannelCmd(1, ENABLE);
}

void TIMER0_IRQHandler(void)
{
    static uint8_t togg = 1;
    if(togg)GPIO_SetValue(PORT_0, RED_LED);
    else GPIO_ClearValue(PORT_0, RED_LED);
    togg = !togg;
    TIM_ClearIntPending(LPC_TIM0, TIM_MR0_INT);
}

void TIMER2_IRQHandler(void)
{
    uint32_t cap = TIM_GetCaptureValue(LPC_TIM2, 0);
    if(GPIO_ReadValue(PORT_0) & ECHO)
    {
        startCap = cap;
    }
    else
    {
        endCap = cap;
        if(endCap >= startCap)distanceTime = endCap - startCap;
    }
}

```

```

        else distanceTime = (0xFFFFFFFF - startCap) + endCap + 1;
//si se pasa
        distance = (distanceTime * 343) / 20000;
        setWaterLevel();
        lowWaterLevel();
        systemUpdate();
    }
    TIM_ClearIntCapturePending(LPC_TIM2, TIM_CR0_INT);
}
void SysTick_Handler(void)//cada 10 us y periodo de 100ms
{
    static uint32_t vTiks = 0;
    if(vTiks < 1) GPIO_SetValue(PORT_0, TRIG);
    else GPIO_ClearValue(PORT_0, TRIG);
    vTiks = (vTiks + 1) % 10000;
    SysTick -> CTRL &= SysTick -> CTRL;
}
//=====
//=====PROGRAMA PRINCIPAL=====
//=====
int main()
{
    cfgPin();
    cfgADC();
    cfgUart();
    cfgTimer0();
    cfgTimer1();
    cfgTimer2();
    SysTick_Config(999);
    NVIC_SetPriority(SysTick_IRQn, 3);
    cfgPWM();
    pumpOff();
    cfgDmaUart();
    while(1){}
    return;
}

```

8.3. HOJAS DE DATOS

Mosfet IRLZ44N

HEXFET® Power MOSFET

- Logic-Level Gate Drive
- Advanced Process Technology
- Dynamic dv/dt Rating
- 175°C Operating Temperature
- Fast Switching
- Fully Avalanche Rated

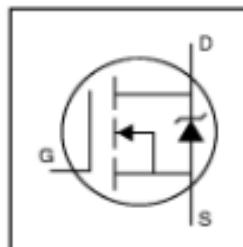
Description

Fifth Generation HEXFETs from International Rectifier utilize advanced processing techniques to achieve the lowest possible on-resistance per silicon area. This benefit, combined with the fast switching speed and ruggedized device design that HEXFET Power MOSFETs are well known for, provides the designer with an extremely efficient device for use in a wide variety of applications.

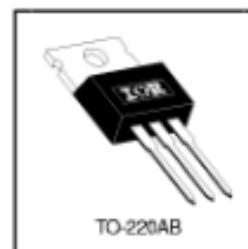
The TO-220 package is universally preferred for all commercial-industrial applications at power dissipation levels to approximately 50 watts. The low thermal resistance and low package cost of the TO-220 contribute to its wide acceptance throughout the industry.

Absolute Maximum Ratings

	Parameter	Max.	Units
$I_D @ T_C = 25^\circ\text{C}$	Continuous Drain Current, $V_{GS} @ 10\text{V}$	47	
$I_D @ T_C = 100^\circ\text{C}$	Continuous Drain Current, $V_{GS} @ 10\text{V}$	33	
I_{DM}	Pulsed Drain Current \textcircled{S}	160	
$P_D @ T_C = 25^\circ\text{C}$	Power Dissipation	110	W
	Linear Derating Factor	0.71	W/ $^\circ\text{C}$
V_{GS}	Gate-to-Source Voltage	± 16	V
E_{AS}	Single Pulse Avalanche Energy \textcircled{S}	210	mJ
I_{AR}	Avalanche Current \textcircled{S}	25	A
E_{AR}	Repetitive Avalanche Energy \textcircled{S}	11	mJ
dv/dt	Peak Diode Recovery dv/dt \textcircled{S}	5.0	V/ns
T_J	Operating Junction and	$-55 \text{ to } +175$	
T_{STG}	Storage Temperature Range	300 (1.6mm from case)	
	Soldering Temperature, for 10 seconds	10 lbf-in (1.1Nm)	
	Mounting torque, 8-32 or M3 screw.		



$V_{DSS} = 55\text{V}$
$R_{DS(on)} = 0.022\Omega$
$I_D = 47\text{A}$



Thermal Resistance

	Parameter	Min.	Typ.	Max.	Units
R_{JC}	Junction-to-Case	—	—	1.4	
R_{CS}	Case-to-Sink, Flat, Greased Surface	—	0.50	—	$^\circ\text{C/W}$
R_{JA}	Junction-to-Ambient	—	—	62	

IRLZ44N

International
I²C Rectifier

Electrical Characteristics @ $T_J = 25^\circ\text{C}$ (unless otherwise specified)

	Parameter	Min.	Typ.	Max.	Units	Conditions
$V_{BR(DSS)}$	Drain-to-Source Breakdown Voltage	55	—	—	V	$V_{DS} = 0\text{V}$, $I_D = 250\mu\text{A}$
$\Delta V_{BR(DSS)}/\Delta T_J$	Breakdown Voltage Temp. Coefficient	—	0.070	—	V/ $^\circ\text{C}$	Reference to 25°C , $I_D = 1\text{mA}$
$R_{D(on)}$	Static Drain-to-Source On-Resistance	—	—	0.022	Ω	$V_{DS} = 10\text{V}$, $I_D = 25\text{A}$ ①
		—	—	0.025	Ω	$V_{DS} = 5.0\text{V}$, $I_D = 25\text{A}$ ②
		—	—	0.035	Ω	$V_{DS} = 4.0\text{V}$, $I_D = 21\text{A}$ ③
$V_{GS(th)}$	Gate Threshold Voltage	1.0	—	2.0	V	$V_{DS} = V_{GS}$, $I_D = 250\mu\text{A}$
g_s	Forward Transconductance	21	—	—	S	$V_{DS} = 25\text{V}$, $I_D = 25\text{A}$
I_{oss}	Drain-to-Source Leakage Current	—	—	25	μA	$V_{DS} = 55\text{V}$, $V_{GS} = 0\text{V}$
I_{oss}	Gate-to-Source Forward Leakage	—	—	100	nA	$V_{DS} = 16\text{V}$
	Gate-to-Source Reverse Leakage	—	—	-100	nA	$V_{DS} = -16\text{V}$
Q_g	Total Gate Charge	—	—	48	nC	$I_D = 25\text{A}$
Q_{gs}	Gate-to-Source Charge	—	—	8.6	nC	$V_{DS} = 44\text{V}$
Q_{gd}	Gate-to-Drain ("Miller") Charge	—	—	25	nC	$V_{DS} = 5.0\text{V}$, See Fig. 6 and 13 ④
$t_{d(on)}$	Turn-On Delay Time	—	11	—	ns	$V_{DD} = 28\text{V}$
t_r	Rise Time	—	84	—	ns	$I_D = 25\text{A}$
$t_{d(off)}$	Turn-Off Delay Time	—	26	—	ns	$R_G = 3.4\Omega$, $V_{GS} = 5.0\text{V}$
t_f	Fall Time	—	15	—	ns	$R_G = 1.1\Omega$, See Fig. 10 ⑤
L_D	Internal Drain Inductance	—	4.5	—	nH	Between lead, 6mm (0.25in.) from package and center of die contact
L_S	Internal Source Inductance	—	7.5	—	nH	
C_{iss}	Input Capacitance	—	1700	—	pF	$V_{GS} = 0\text{V}$
C_{oss}	Output Capacitance	—	400	—	pF	$V_{DS} = 25\text{V}$
C_{rdss}	Reverse Transfer Capacitance	—	150	—	pF	$f = 1.0\text{MHz}$, See Fig. 5

Source-Drain Ratings and Characteristics

	Parameter	Min.	Typ.	Max.	Units	Conditions
I_S	Continuous Source Current (Body Diode)	—	—	47	A	MOSFET symbol showing the integral reverse p-n junction diode.
I_{SM}	Pulsed Source Current (Body Diode) ⑥	—	—	160	A	
V_{SD}	Diode Forward Voltage	—	—	1.3	V	$T_J = 25^\circ\text{C}$, $I_S = 25\text{A}$, $V_{GS} = 0\text{V}$ ⑦
t_{rr}	Reverse Recovery Time	—	80	120	ns	$T_J = 25^\circ\text{C}$, $I_S = 25\text{A}$
Q_{rr}	Reverse Recovery Charge	—	210	320	nC	$dI/dt = 100\text{A}/\mu\text{s}$ ⑧
t_{on}	Forward Turn-On Time	Intrinsic turn-on time is negligible (turn-on is dominated by L_S+L_D)				

Notes:

- ① Repetitive rating; pulse width limited by max. junction temperature. (See fig. 11)
- ② $V_{DD} = 25\text{V}$, starting $T_J = 25^\circ\text{C}$, $L = 470\mu\text{H}$, $R_G = 250\Omega$, $I_{GS} = 25\text{A}$. (See Figure 12)
- ③ $I_{SD} \leq 25\text{A}$, $dI/dt \leq 270\text{A}/\mu\text{s}$, $V_{DD} \leq V_{BR(DSS)}$, $T_J \leq 175^\circ\text{C}$
- ④ Pulse width $\leq 300\mu\text{s}$; duty cycle $\leq 2\%$.

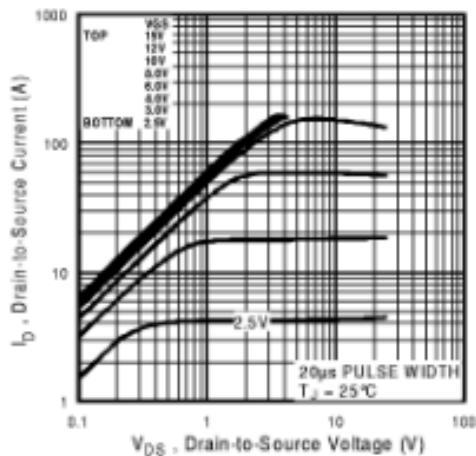


Fig 1. Typical Output Characteristics

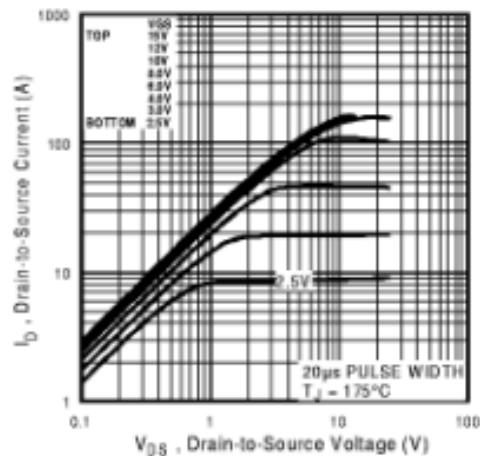


Fig 2. Typical Output Characteristics

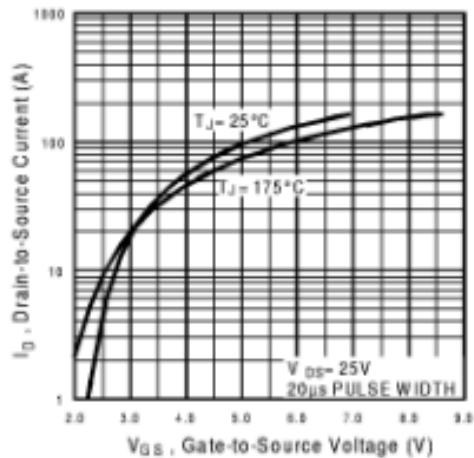


Fig 3. Typical Transfer Characteristics

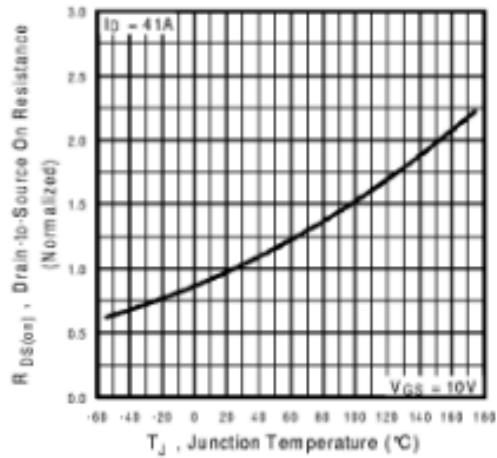


Fig 4. Normalized On-Resistance
Vs. Temperature

CAPACITIVE MOISTURE SENSOR



Features

1. New soil moisture sensor. This capacitive soil moisture sensor is different from most resistive sensors on the market. It uses the principle of capacitive sensing to detect soil moisture. The problem that the resistance sensor is easily corroded is avoided, and its working life is greatly extended.
2. The sensor has a built-in voltage regulator chip, which supports a 3.3~5.5V wide voltage working environment, which means that it can work normally even on the 3.3V Arduino main control board. The iconic DFRobot-Gravity interface ensures the compatibility of the interface and can be directly connected to the Gravity IO expansion board.
3. A micro PC such as a Raspberry Pi only needs an external ADC (analog signal to digital signal) conversion module to work.
4. With an external screen and a motherboard, you can talk to your plant to see if the beloved one is thirsty, and whether it needs a little more water.

Product parameters

Working voltage: 3.3 ~ 5.5 VDC

Output voltage: 0 ~ 3.0 VDC

Interface: PH2.54-3P

Size: 98 x 23mm (LxW)

Instructions

Prepare

Hardware

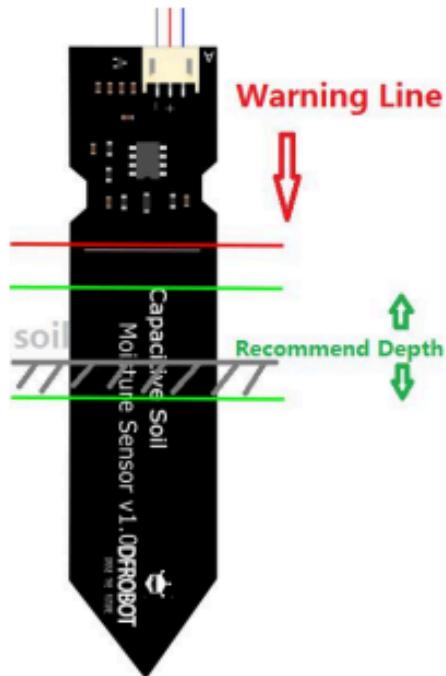
- UNO control board x1
- Soil moisture sensor x1
- PH2.54-3P wiring x1

Software

- Arduino IDE V1.6.5

Wiring diagram

1. Connect the sensor and the main control board as shown



Interval setting

Because the sensor value will be affected by the depth of the soil and the lightness of the soil, only the relative humidity of the soil can be detected. We divide the range of humidity into three equal parts, which means dry, humid, and very humid. The two data recorded before are the humidity interval. For example: the reading in the air is 520, and the reading in the water is 260, so it can be divided into (520,430), (430,350), (350,260). These three sections represent dry, wet, and very humid.

Note: Since this sensor will monitor soil moisture based on the principle of capacitive sensing, placing it in different places with different soil moisture, different lightness, and different insertion depth will reflect different humidity, even in the same place, at the same depth, at During the second insertion, since the first extraction has caused loosening of the soil, the humidity may be lower than the first reading. A

Note: Humidity is inversely proportional to the reading.

HC-SR04

HC-SR04 Ultrasonic Sensor

Elijah J. Morgan

Nov. 16 2014

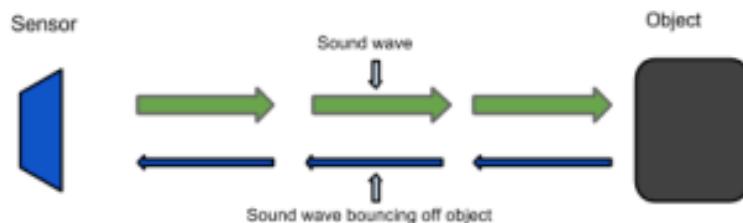
The purpose of this file is to explain how the HC-SR04 works. It will give a brief explanation of how ultrasonic sensors work in general. It will also explain how to wire the sensor up to a microcontroller and how to take/interpret readings. It will also discuss some sources of errors and bad readings.

1. How Ultrasonic Sensors Work
2. HC-SR04 Specifications
3. Timing chart, Pin explanations and Taking Distance Measurements
4. Wiring HC-SR04 with a microcontroller
5. Errors and Bad Readings



1. How Ultrasonic Sensors Work

Ultrasonic sensors use sound to determine the distance between the sensor and the closest object in its path. How do ultrasonic sensors do this? Ultrasonic sensors are essentially sound sensors, but they operate at a frequency above human hearing.



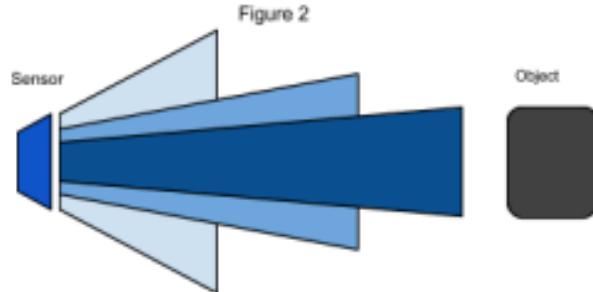
The sensor sends out a sound wave at a specific frequency. It then listens for that specific sound wave to bounce off of an object and come back (Figure 1). The sensor keeps track of the time between sending the sound wave and the sound wave returning. If you know how fast something is going and how long it is traveling you can find the distance traveled with equation 1.

$$\text{Equation 1. } d = v \times t$$

The speed of sound can be calculated based on the a variety of atmospheric conditions, including temperature, humidity and pressure. Actually calculating the distance will be shown later on in this document.

It should be noted that ultrasonic sensors have a cone of detection, the angle of this cone varies with distance, Figure 2 show this relation. The ability of a sensor to

detect an object also depends on the objects orientation to the sensor. If an object doesn't present a flat surface to the sensor then it is possible the sound wave will bounce off the object in a way that it does not return to the sensor.



2. HC-SR04 Specifications

The sensor chosen for the Firefighting Drone Project was the HC-SR04. This section contains the specifications and why they are important to the sensor module. The sensor modules requirements are as follows.

- Cost
- Weight
- Community of hobbyists and support
- Accuracy of object detection
- Probability of working in a smoky environment
- Ease of use

The HC-SR04 Specifications are listed below. These specifications are from the Cytron Technologies HC-SR04 User's Manual (source 1).

- Power Supply: +5V DC
- Quiescent Current: <2mA
- Working current: 15mA
- Effectual Angle: <15°
- Ranging Distance: 2-400 cm
- Resolution: 0.3 cm
- Measuring Angle: 30°
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm
- Weight: approx. 10 g

The HC-SR04's best selling point is its price; it can be purchased at around \$2 per unit.

HC-05

HC-05

-Bluetooth to Serial Port Module

Overview



HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.

Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH(Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.

Specifications

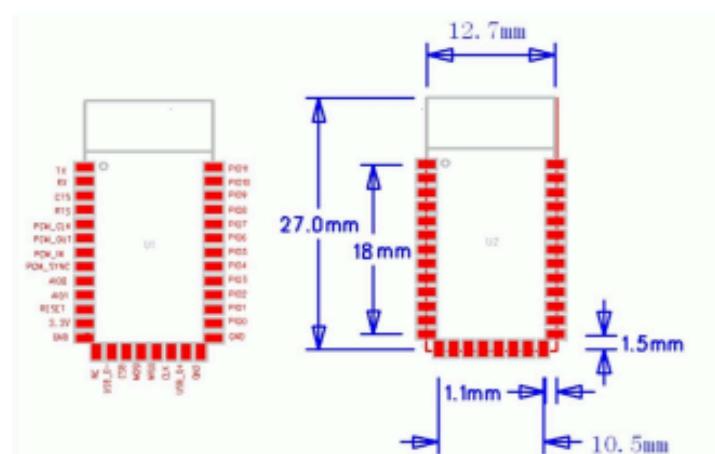
Hardware features

- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

Software features

- Default Baud rate: 38400, Data bits:8, Stop bit:1, Parity:No parity, Data control: has. Supported baud rate: 9600, 19200, 38400, 57600, 115200, 230400, 460800.
- Given a rising pulse in PIO0, device will be disconnected.
- Status instruction port PIO1: low-disconnected, high-connected;
- PIO10 and PIO11 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"0000" as default
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

Hardware



PIN Name	PIN #	Pad type	Description	Note
GND	13 21 22	VSS	Ground pin	
3.3 VCC	12	3.3V	Integrated 3.3V (+) supply with On-chip linear regulator output within 3.15-3.3V	
AIO0	9	Bi-Directional	Programmable input/output line	
AIO1	10	Bi-Directional	Programmable input/output line	
PIO0	23	Bi-Directional RX EN	Programmable input/output line, control output for LNA(if fitted)	
PIO1	24	Bi-Directional TX EN	Programmable input/output line, control output for PA(if fitted)	
PIO2	25	Bi-Directional	Programmable input/output line	
PIO3	26	Bi-Directional	Programmable input/output line	
PIO4	27	Bi-Directional	Programmable input/output line	
PIO5	28	Bi-Directional	Programmable input/output line	
PIO6	29	Bi-Directional	Programmable input/output line	
PIO7	30	Bi-Directional	Programmable input/output line	
PIO8	31	Bi-Directional	Programmable input/output line	
PIO9	32	Bi-Directional	Programmable input/output line	
PIO10	33	Bi-Directional	Programmable input/output line	
PIO11	34	Bi-Directional	Programmable input/output line	

R385 Diaphragm Mini Water Pump 12VDC



New advanced version with high lift self priming diaphragm water pump. This water pump has more lift power than the RS-360, bigger outlets diameters, and the rubber retainers helps to secure the motor to some stationary object. This motor is more suitable for practical needs, than other toy water motor pumps. This pump cannot be put in water (liquid) when using, not submersible.

FEATURES:

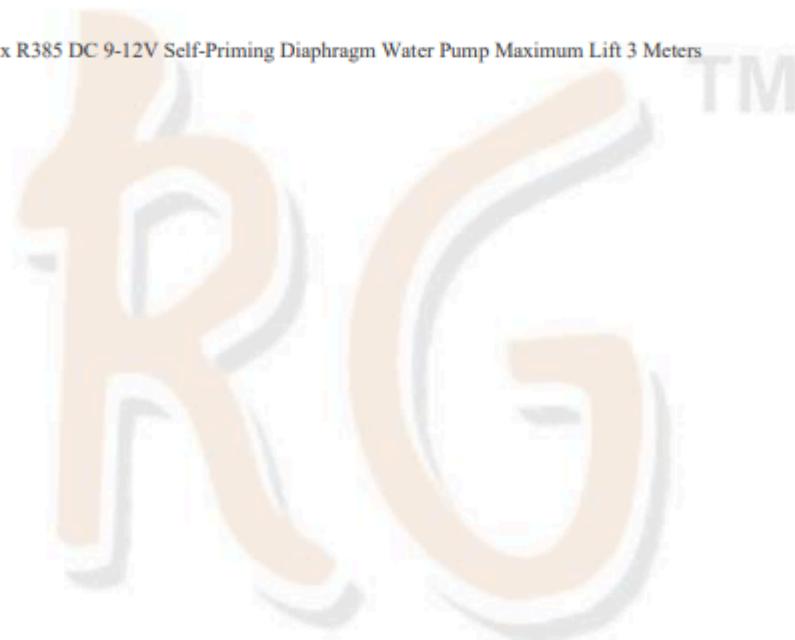
- Model Number: R-385
- Rated Voltage: DC 9 V to 12V (1 amps)
- Load current: 0.7A (Max)
- Flow: 1.5 to 2 Ltr/min
- Power: 4W-7W
- Max Lift: 3m
- Max Suction: 2m
- Max Water Temp: 70 °C
- CPump Size: 90mm * 40mm * 35mm approx
- Fluid: 0-100 ° C

Rajguru Electronics (I) Pvt. Ltd.

- Input/output tube diameter: outer 8.5mm, inner 6mm approx
- Max Current: Up to 2 Amps while starting up
- Life: up to 2500 Hours

PACKAGE INCLUDES:

1 x R385 DC 9-12V Self-Priming Diaphragm Water Pump Maximum Lift 3 Meters



Bc548

HN / BC 546...549

NPN Silicon Epitaxial Planar Transistor

These transistors are subdivided into three groups A, B and C according to their current gain. The type BC546 is available in groups A and B, however, the types BC547 and BC548 can be supplied in all three groups. The BC549 is a low-noise type and available in groups B and C. As complementary types, the PNP transistors BC556...BC559 are recommended.

On special request, these transistors can be manufactured in different pin configurations. Please refer to the "TO-92 TRANSISTOR PACKAGE OUTLINE" on page 80 for the available pin options.



TO-92 Plastic Package
Weight approx. 0.18 g
Dimensions in mm

Absolute Maximum Ratings ($T_A = 25^\circ\text{C}$)

		Symbol	Value	Unit
Collector-Base Voltage	HN / BC 546	V_{CEO}	80	V
	HN / BC 547	V_{CEO}	50	V
	HN / BC 548, HN / BC 549	V_{CEO}	30	V
Collector-Emitter Voltage	HN / BC 546	V_{CES}	85	V
	HN / BC 547	V_{CES}	50	V
	HN / BC 548, HN / BC 549	V_{CES}	30	V
Collector-Emitter Voltage	HN / BC 546	V_{CEO}	65	V
	HN / BC 547	V_{CEO}	45	V
	HN / BC 548, HN / BC 549	V_{CEO}	30	V
Emitter-Base Voltage	HN / BC 546, HN / BC 547	V_{ESO}	6	V
	HN / BC 548, HN / BC 549	V_{ESO}	5	V
Collector Current		I_C	100	mA
Peak Collector Current		I_{CM}	200	mA
Peak Base Current		I_{BM}	200	mA
Peak Emitter Current		$-I_{EM}$	200	mA
Power Dissipation at $T_{J\max} = 25^\circ\text{C}$		P_{diss}	500 ^{II}	mW
Junction Temperature		T_J	150	$^\circ\text{C}$
Storage Temperature Range		T_S	-65 to +150	$^\circ\text{C}$

^{II}Valid provided that leads are kept at ambient temperature at a distance of 2 mm from case

G S P FORM A AVAILABLE



SEMTECH ELECTRONICS LTD.

(wholly owned subsidiary of HONEY TECHNOLOGY LTD.)



HN / BC 546...549

Characteristics at $T_{\text{amb}} = 25^{\circ}\text{C}$

	Symbol	Min.	Typ.	Max.	Unit
h-Parameters at $V_{\text{CE}} = 5\text{V}$, $I_c = 2\text{mA}$, $f = 1\text{kHz}$, Small Signal Current Gain	h_{v} h_{v} h_{v}	- - -	220 330 600	- - -	-
Input Impedance	Current Gain Group A h_{v} h_{v} h_{v}	1.6 3.2 8	2.7 4.5 8.7	4.5 8.5 15	$\text{k}\Omega$
Output Admittance	Current Gain Group A h_{v} h_{v} h_{v}	- - -	18 30 60	30 60 110	μs
Reverse Voltage Transfer Ratio	Current Gain Group A h_{v} h_{v} h_{v}	- - -	$1.5 \cdot 10^{-4}$ $2 \cdot 10^{-4}$ $3 \cdot 10^{-4}$	- - -	-
DC Current Gain, at $V_{\text{CE}} = 5\text{V}$, $I_c = 10\text{\mu A}$	h_{ve} h_{ve} h_{ve}	- - -	90 150 270	- - -	-
at $V_{\text{CE}} = 5\text{V}$, $I_c = 2\text{mA}$	Current Gain Group A h_{ve} h_{ve} h_{ve}	110 200 420	180 290 500	220 450 800	-
at $V_{\text{CE}} = 5\text{V}$, $I_c = 100\text{mA}$	Current Gain Group A h_{ve} h_{ve} h_{ve}	- - -	120 200 400	- - -	-
Thermal Resistance Junction to Ambient Air	R_{JA}	-	-	250 ⁽¹⁾	K/W
Collector Saturation Voltage at $I_c = 10\text{mA}$, $I_b = 0.5\text{mA}$ at $I_c = 100\text{mA}$, $I_b = 5\text{mA}$	V_{CEsat} V_{Bsat}	- -	80 200	200 600	mV mV
Base Saturation Voltage at $I_c = 10\text{mA}$, $I_b = 0.5\text{mA}$ at $I_c = 100\text{mA}$, $I_b = 5\text{mA}$	V_{Bsat} V_{Bsat}	- -	700 900	- -	mV mV
Base Emitter Voltage at $V_{\text{CE}} = 5\text{V}$, $I_c = 2\text{mA}$ at $V_{\text{CE}} = 5\text{V}$, $I_c = 10\text{mA}$	V_{BE} V_{BE}	580 -	660 -	700 720	mV mV
Collector Emitter Cutoff Current at $V_{\text{CE}} = 80\text{V}$ HN / BC 546 at $V_{\text{CE}} = 50\text{V}$ HN / BC 547	I_{CES} I_{CES}	- -	0.2 0.2	15 15	nA nA
at $V_{\text{CE}} = 30\text{V}$ HN / BC 548, HN / BC 549	I_{CES}	-	0.2	15	nA
at $V_{\text{CE}} = 80\text{V}$, $T_j = 125^{\circ}\text{C}$ HN / BC 546 at $V_{\text{CE}} = 50\text{V}$, $T_j = 125^{\circ}\text{C}$ HN / BC 547	I_{CES} I_{CES}	- -	- -	4 4	μA μA

⁽¹⁾ Valid provided that leads are kept at ambient temperature at a distance of 2 mm from case.



SEMTECH ELECTRONICS LTD.

(wholly owned subsidiary of HONEY TECHNOLOGY LTD.)



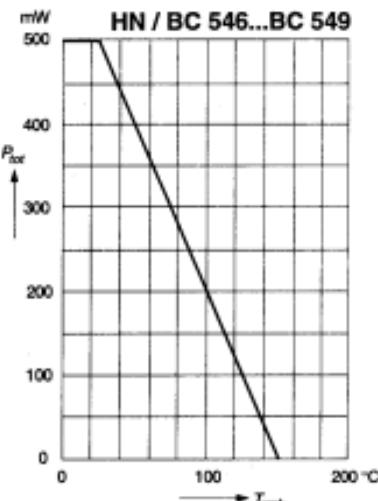
HN / BC 546...549

Characteristics, continuation

	Symbol	Min.	Typ.	Max.	Unit
at $V_{CE} = 30V$, $T_i = 125^\circ C$ HN / BC 548, HN / BC 549	I_{CES}	-	-	4 4	μA μA
Gain-Bandwidth Product at $V_{CE} = 5V$, $I_C = 10mA$, $f = 100MHz$	f_T	-	300	-	MHz
Collector-Base Capacitance at $V_{CE} = 10V$, $f = 1MHz$	C_{CBO}	-	3.5	6	pF
Emitter-Base Capacitance at $V_{CE} = 0.5V$, $f = 1MHz$	C_{EBO}	-	9	-	pF
Noise Figure at $V_{CE} = 5V$, $I_C = 200\mu A$, $R_E = 2k\Omega$, $f = 1kHz$, $\Delta f = 200Hz$ HN / BC 546, HN / BC 547	F	-	2	10	dB
HN / BC 548 HN / BC 549	F	-	1.2	4	dB

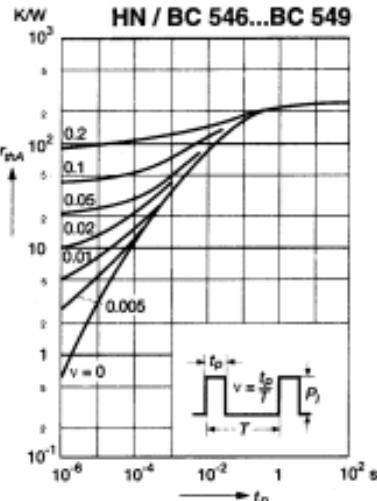
Admissible power dissipation versus temperature

Valid provided that leads are kept at ambient temperature at a distance of 2 mm from case



Pulse thermal resistance versus pulse duration

Valid provided that leads are kept at ambient temperature at a distance of 2 mm from case

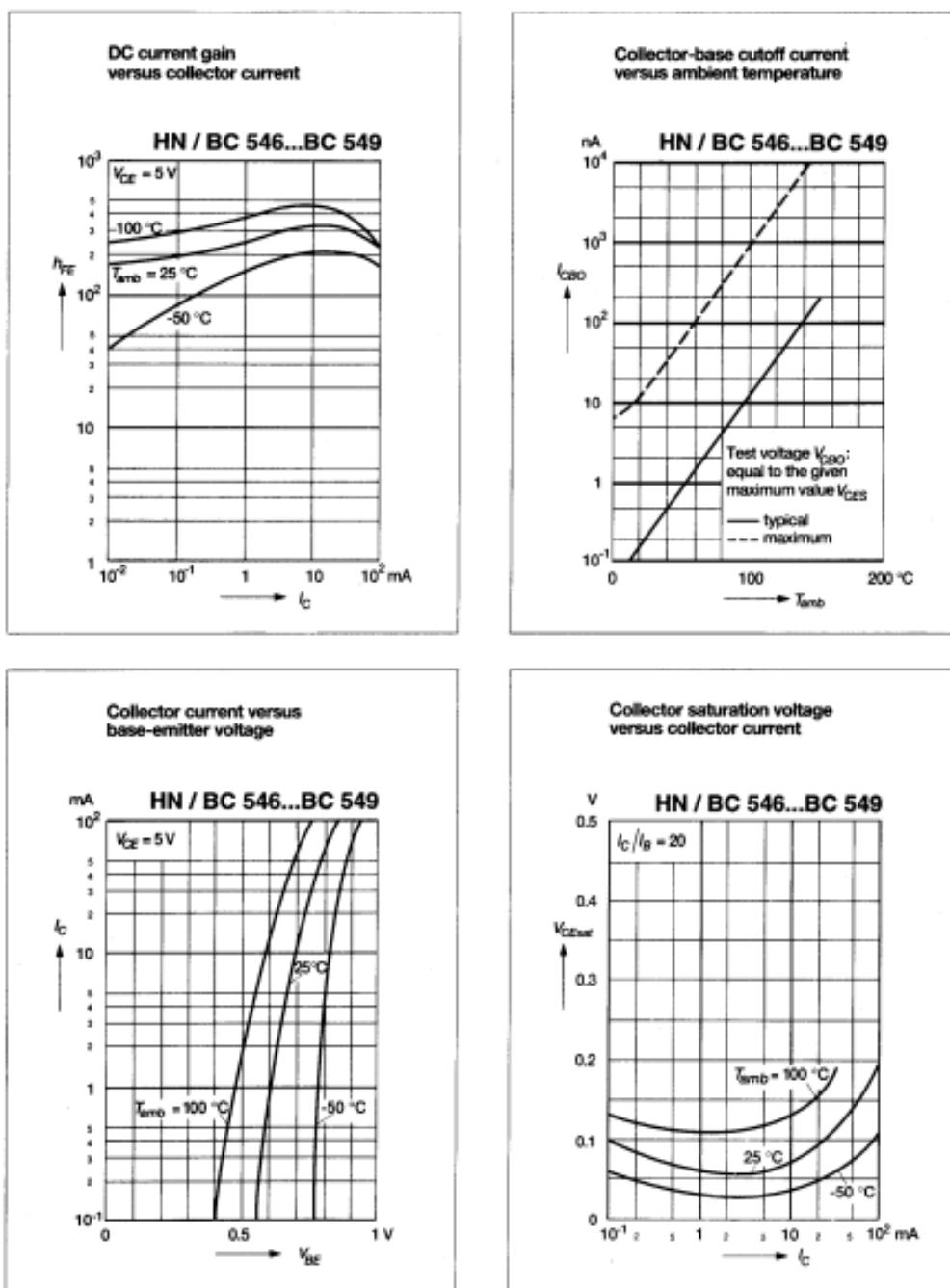


SEMTECH ELECTRONICS LTD.

(wholly owned subsidiary of **HONEY TECHNOLOGY LTD.**)



HN / BC 546...549



SEMTECH ELECTRONICS LTD.

(wholly owned subsidiary of HONEY TECHNOLOGY LTD.)



FIN