Anexos

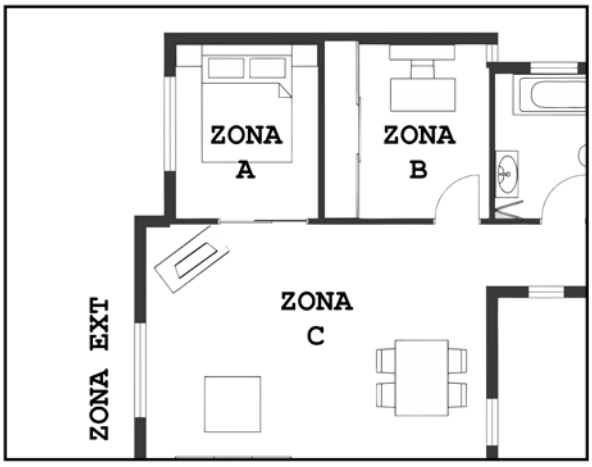
Anexo I. Modelado térmico de viviendas mediante redes neuronales

Se propone un esquema a grandes rasgos para utilizar los datos recogidos y entrenar redes neuronales artificiales capaces de estimar temperaturas futuras. Su potencial aplicación es el control de sistemas de calefacción y aire acondicionado, buscando eficiencia energética y confort.

Para mejorar el funcionamiento de los sistemas de calefacción se propone una aproximación a la resolución de la función de la temperatura en una zona como la interconexión de las temperaturas previas, las temperaturas de los cuartos colindantes, humedades de los mismos, estado de la calefacción, temperatura externa, presión atmosférica, velocidad del viento, radiación UV…

$$t_{zona\ A} = f(t_A^{-1}, t_B^{-1}, \dots, t_N^{-1}, H_A^{-1}, H_B^{-1}, \dots, H_N^{-1}, z\ parámetros\ ambientales)$$

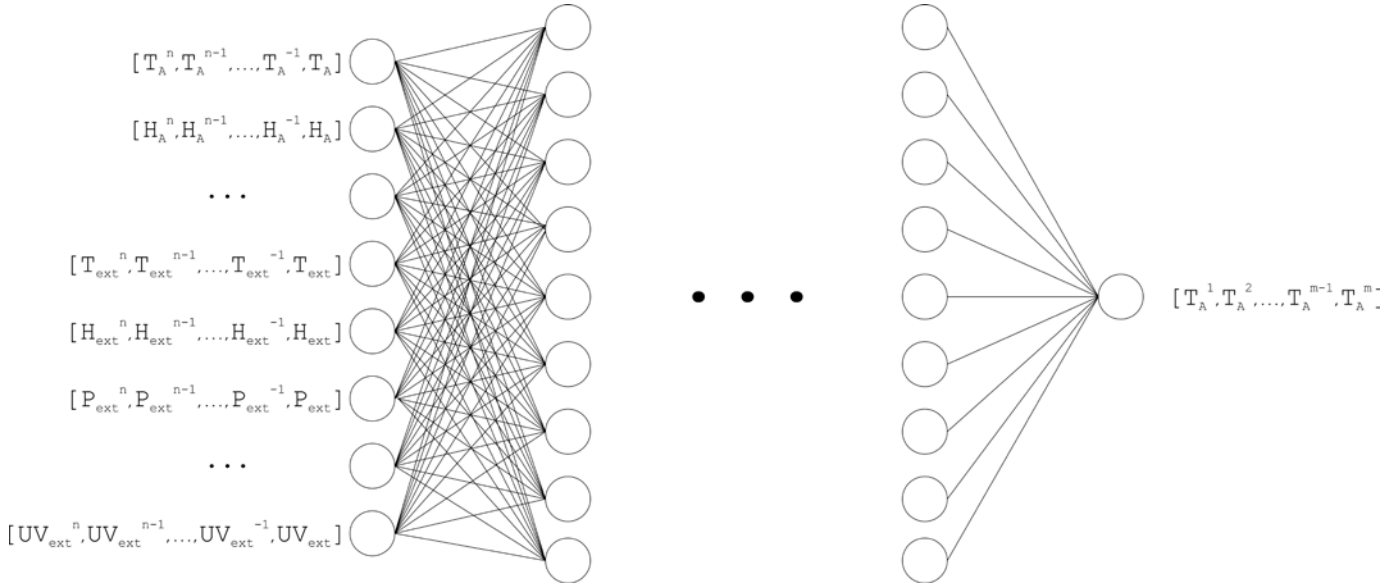
Hay que tener en cuenta, que para cada zona de estudio, se deberán considerar las variables de las zonas colindantes que afecten directamente a la temperatura de este. Por ejemplo, en la siguiente ilustración, la temperatura de la zona A dependerá directamente de las variables en las zonas B, C y la zona externa.



Esta función se podría resolver mediante el sistema de redes neuronales, planteando, por ejemplo, una red tipo perceptrón multicapa (aproximador universal de funciones) [1] completamente interconectada, donde la capa de entrada está constituida por vectores de las z variables contempladas con registros temporales previos discretizados desde n momentos anteriores (*Temporal Delayed Neural Network)* [1]. Habría que determinar todos los factores de la red como el número de capas ocultas o la función de activación de las mismas. Otra de las entradas que han de ser contempladas es el estado de los elementos de calefacción/climatización de la vivienda en los instantes temporales, ya que son elementos que pueden perturbar la distribución natural de calor en la vivienda.

La red debería ser capaz de parametrizar el problema y ser capaz de modelar una estimación de las m temperaturas siguientes del cuarto estudiado.

El funcionamiento de la red propuesta es el siguiente:



El prototipo desarrollado en el trabajo sería relevante para elaborar los vectores de datos de entrada y también para el entrenamiento de la misma. Se plantea un aprendizaje supervisado. Es decir, para definir los pesos y *bias* de cada una de las neuronas, se requiere un entrenamiento previo en base a datos históricos. Se ha de recoger valores de todas las variables contempladas durante un lapso temporal representativo, como mínimo un año.

Otra opción menos *visual*, podría ser plantear una red neuronal donde todas las variables de la vivienda sean introducidas como variables de entrada; las temperaturas y humedades de todos los cuartos, todas las variables externas, y el estado de los elementos de climatización. Como salida de la red se plantearían n neuronas correspondientes a las n zonas de las que se quiere una estimación de la temperatura. De esta manera no se requeriría la elaboración de una red por cada zona a controlar, sino que con una única red se podría predecir el comportamiento de todas las zonas de la vivienda.

Por supuesto, todo esto es un planteamiento teórico y en el que se proponen técnicas muy básicas. Posiblemente existan técnicas de aprendizaje o estructuras de red mucho más adecuadas para este problema, que deberán ser estudiadas.

[1]: Bonifacio Martín-del-Brío y Alfredo Sanz, Redes Neuronales y Sistemas Borrosos, Editorial RA-MA, Madrid y Alfaomenga, Mexico DF, 2006

Anexo II. Datasheet Microprocesador QG8

Chapter 1
Device Overview

1.1 Introduction

The MC9S08QG8 is a member of the low-cost, high-performance HCS08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced HCS08 core and are available with a variety of modules, memory sizes, memory types, and package types. Refer to Table 1-1 for features associated with each device in this series.

1.1.1 Devices in the MC9S08QG8/4 Series

Table 1-1 summarizes the features available in the MC9S08QG8/4 series of MCUs.

Table 1-1. Devices in the MC9S08QG8/4 Series

Feature	Device					
	MC9S08QG8			MC9S08QG4		
Package	24-Pin	16-Pin	8-Pin	24-Pin	16-Pin	8-Pin
FLASH	8K			4K		
RAM	512			256		
XOSC	yes	yes	no	yes	yes	no
ICS	yes			yes		
ACMP	yes			yes		
ADC	8-ch	8-ch	4-ch	8-ch	8-ch	4-ch
DBG	yes			yes	yes	yes
IIC	yes			yes		
IRQ	yes			yes		
KBI	8-pin	8-pin	4-pin	8-pin	8-pin	4-pin
MTIM	yes			yes		
SCI	yes	yes	no	yes	yes	no
SPI	yes	yes	no	yes	yes	no
TPM	2-ch	2-ch	1-ch	2-ch	2-ch	1-ch
I/O pins	12 I/O 1 Output only 1 Input only	12 I/O 1 Output only 1 Input only	4 I/O 1 Output only 1 Input only	12 I/O 1 Output only 1 Input only	12 I/O 1 Output only 1 Input only	4 I/O 1 Output only 1 Input only
Package Types	24 QFN	16 PDIP 16 QFN 16 TSSOP	8 DFN 8 SOIC	24 QFN	16 QFN 16 TSSOP	8 DFN 8 PDIP 8 SOIC

3.6 Stop Modes

One of three stop modes is entered upon execution of a STOP instruction when STOPE in SOPT1 is set. In any stop mode, the bus and CPU clocks are halted. The ICS module can be configured to leave the reference clocks running. See Chapter 10, "Internal Clock Source (S08ICSV1)," for more information.

Table 3-1 shows all of the control bits that affect stop mode selection and the mode selected under various conditions. The selected mode is entered following the execution of a STOP instruction.

Table 3-1. Stop Mode Selection

STOPE	ENBDM [1]	LVDE	LVDSE	PDC	PPDC	Stop Mode
0	x	x		x	x	Stop modes disabled; illegal opcode reset if STOP instruction executed
1	1	x		x	x	Stop3 with BDM enabled [2]
1	0	Both bits must be 1		x	x	Stop3 with voltage regulator active
1	0	Either bit a 0		0	x	Stop3
1	0	Either bit a 0		1	1	Stop2
1	0	Either bit a 0		1	0	Stop1

[1] ENBDM is located in the BDCSCR which is only accessible through BDC commands; see Section 17.4.1.1, "BDC Status and Control Register (BDCSCR)".

[2] When in Stop3 mode with BDM enabled, the S_{IDD} will be near R_{IDD} levels because internal clocks are enabled.

3.6.1 Stop3 Mode

Stop3 mode is entered by executing a STOP instruction under the conditions as shown in Table 3-1. The states of all of the internal registers and logic, RAM contents, and I/O pin states are maintained.

Stop3 can be exited by asserting $\overline{\text{RESET}}$, or by an interrupt from one of the following sources: the real-time interrupt (RTI), LVD, ADC, $\overline{\text{IRQ}}$, or the KBI.

If stop3 is exited by means of the $\overline{\text{RESET}}$ pin, then the MCU is reset and operation will resume after taking the reset vector. Exit by means of one of the internal interrupt sources results in the MCU taking the appropriate interrupt vector.

3.6.1.1 LVD Enabled in Stop Mode

The LVD system is capable of generating either an interrupt or a reset when the supply voltage drops below the LVD voltage. If the LVD is enabled in stop (LVDE and LVDSE bits in SPMSC1 both set) at the time the CPU executes a STOP instruction, then the voltage regulator remains active during stop mode.

For the ADC to operate the LVD must be left enabled when entering stop3.

3.6.1.2 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if ENBDM in BDCSCR is set. This register is described in Chapter 17, "Development Support." If ENBDM is set when the CPU executes a

Chapter 4
Memory Map and Register Definition

4.1 MC9S08QG8/4 Memory Map

As shown in Figure 4-1, on-chip memory in the MC9S08QG8/4 series of MCUs consists of RAM, FLASH program memory for nonvolatile data storage, and I/O and control/status registers. The registers are divided into these groups:

- Direct-page registers (0x0000 through 0x005F)
- High-page registers (0x1800 through 0x184F)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

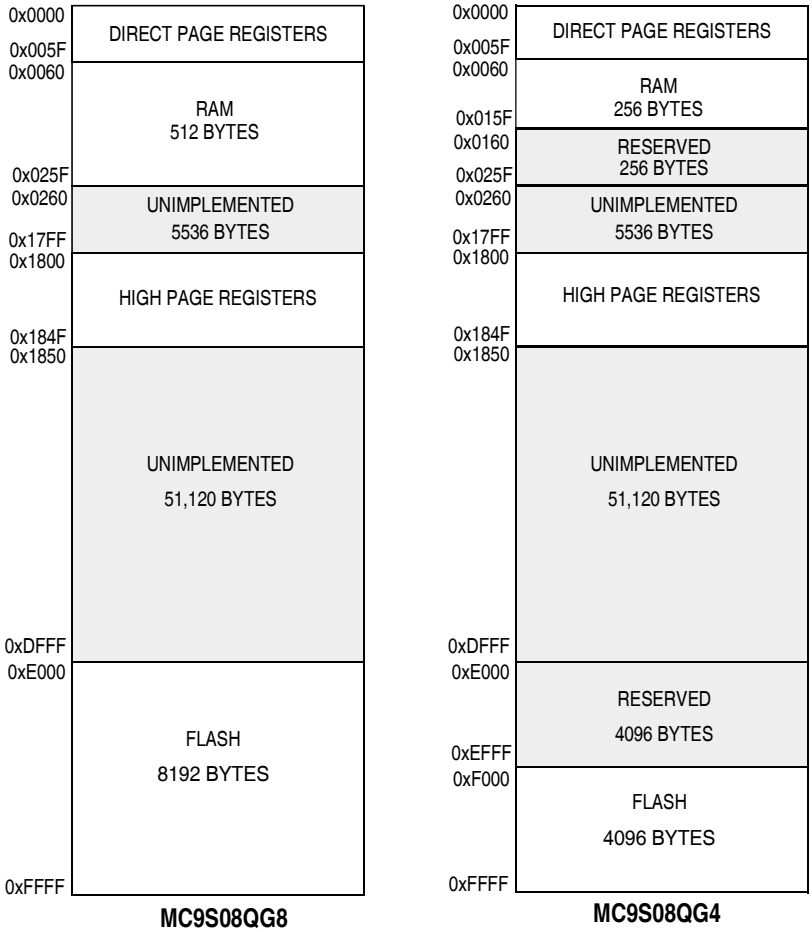


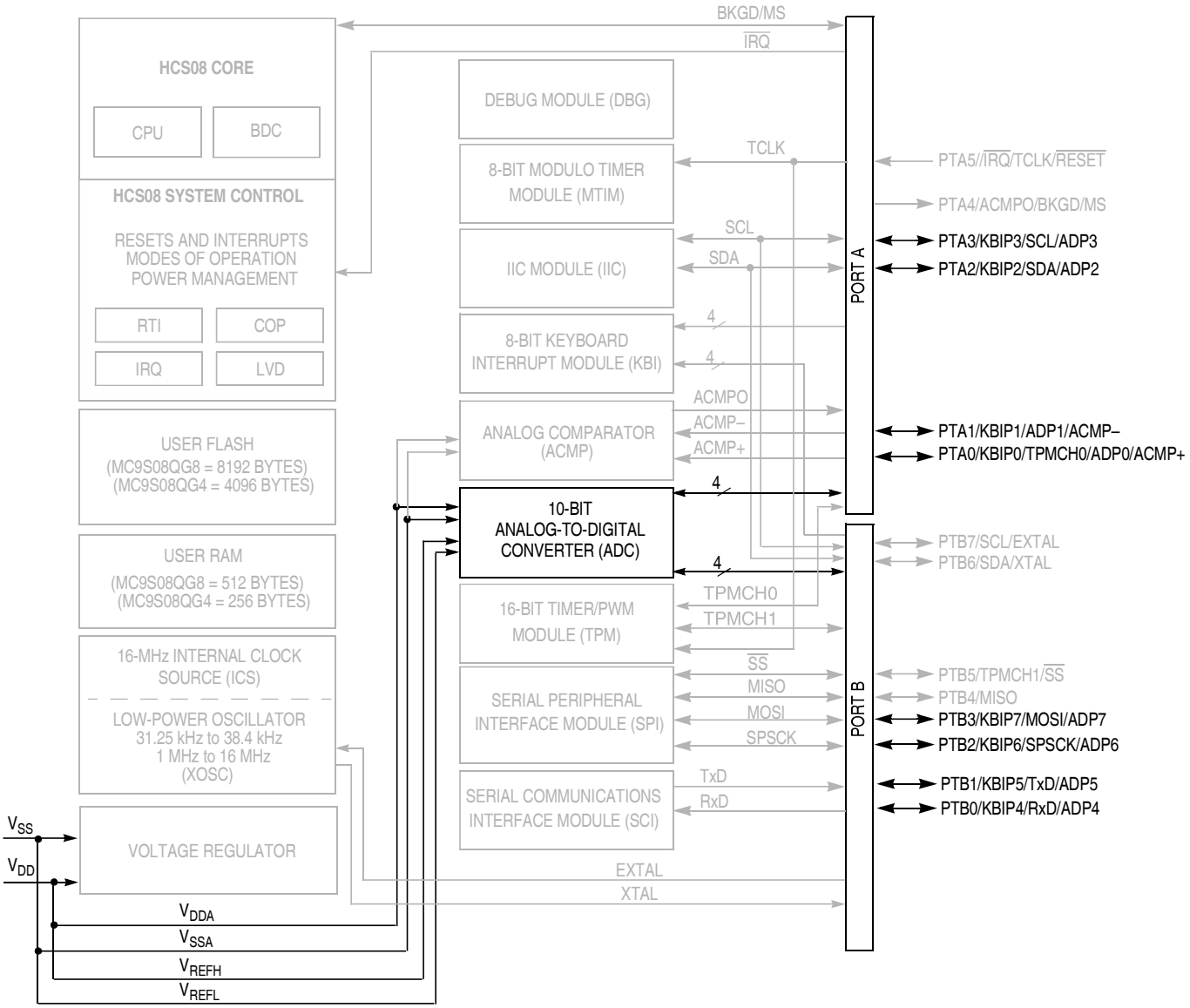
Figure 4-1. MC9S08QG8/4 Memory Map

Chapter 9
Analog-to-Digital Converter (S08ADC10V1)

9.1 Introduction

The 10-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

Figure 9-1 shows the MC9S08QG8/4 with the ADC module and pins highlighted.



NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. IRQ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as IRQ pin function (IRQPE = 1).
6. $\overline{\text{RESET}}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 9-1. MC9S08QG8/4 Block Diagram Highlighting ADC Block and Pins

9.1.1 Module Configurations

This section provides device-specific information for configuring the ADC on MC9S08QG8/4 devices.

9.1.1.1 Analog Supply and Voltage Reference Connections

The V_{DDAD} and V_{REFH} sources for the ADC are internally connected to the V_{DD} pin. The V_{SSAD} and V_{REFL} sources for the ADC are internally connected to the V_{SS} pin.

9.1.1.2 Channel Assignments

The ADC channel assignments for the MC9S08QG8/4 devices are shown in Table 9-1. Reserved channels convert to an unknown value.

Table 9-1. ADC Channel Assignment

ADCH	Channel	Input	Pin Control
00000	AD0	PTA0/ADP0	ADPC0
00001	AD1	PTA1/ADP1	ADPC1
00010	AD2	PTA2/ADP2	ADPC2
00011	AD3	PTA3/ADP3	ADPC3
00100	AD4	PTB0/ADP4	ADPC4
00101	AD5	PTB1/ADP5	ADPC5
00110	AD6	PTB2/ADP6	ADPC6
00111	AD7	PTB3/ADP7	ADPC7
01000	AD8	V_{SS}	N/A
01001	AD9	V_{SS}	N/A
01010	AD10	V_{SS}	N/A
01011	AD11	V_{SS}	N/A
01100	AD12	V_{SS}	N/A
01101	AD13	V_{SS}	N/A
01110	AD14	V_{SS}	N/A
01111	AD15	V_{SS}	N/A

ADCH	Channel	Input	Pin Control
10000	AD16	V_{SS}	N/A
10001	AD17	V_{SS}	N/A
10010	AD18	V_{SS}	N/A
10011	AD19	V_{SS}	N/A
10100	AD20	V_{SS}	N/A
10101	AD21	V_{SS}	N/A
10110	AD22	Reserved	N/A
10111	AD23	Reserved	N/A
11000	AD24	Reserved	N/A
11001	AD25	Reserved	N/A
11010	AD26	Temperature Sensor[1]	N/A
11011	AD27	Internal Bandgap	N/A
11100	—	Reserved	N/A
11101	V_{REFH}	V_{DD}	N/A
11110	V_{REFL}	V_{SS}	N/A
11111	Module Disabled	None	N/A

1 For information, see Section 9.1.1.6, “Temperature Sensor.”

NOTE

Selecting the internal bandgap channel requires BGBE =1 in SPMSC1; see Section 5.8.8, “System Power Management Status and Control 1 Register (SPMSC1).” For the value of the bandgap voltage reference see Section A.5, “DC Characteristics.”

9.1.1.3 Alternate Clock

The ADC is capable of performing conversions using the MCU bus clock, the bus clock divided by two, or the local asynchronous clock (ADACK) within the module. The alternate clock, ALTCLK, input for the MC9S08QG8/4 MCU devices is not implemented.

9.1.1.4 Hardware Trigger

The ADC hardware trigger, ADHWT, is output from the real-time interrupt (RTI) counter. The RTI counter can be clocked by either ICSERCLK or a nominal 1-kHz clock source within the RTI block.

The period of the RTI is determined by the input clock frequency and the RTIS bits. The RTI counter is a free running counter that generates an overflow at the RTI rate determined by the RTIS bits. When the ADC hardware trigger is enabled, a conversion is initiated upon an RTI counter overflow.

The RTI can be configured to cause a hardware trigger in MCU run, wait, and stop3.

9.1.1.5 Analog Pin Enables

The ADC on MC9S08QG8 devices contains only one analog pin enable register, APCTL1.

9.1.1.6 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 9-1 provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{TEMP} - V_{TEMP25}) \div m) \qquad \textit{Eqn. 9-1}$$

where:

- V_{TEMP} is the voltage of the temperature sensor channel at the ambient temperature.
- V_{TEMP25} is the voltage of the temperature sensor channel at 25°C.
- m is the hot or cold voltage versus temperature slope in V/°C.

For temperature calculations, use the V_{TEMP25} and m values from Section A.10, “ADC Characteristics,” in Appendix A, “Electrical Characteristics.”

In application code, the user reads the temperature sensor channel, calculates V_{TEMP}, and compares to V_{TEMP25}. If V_{TEMP} is greater than V_{TEMP25}, the cold slope value is applied in Equation 9-1. If V_{TEMP} is less than V_{TEMP25} the hot slope value is applied in Equation 9-1.

9.1.1.7 Low-Power Mode Operation

The ADC is capable of running in stop3 mode but requires LVDSE and LVDE in SPMSC1 to be set.

9.1.2 Features

Features of the ADC module include:

- Linear successive approximation algorithm with 10 bits resolution.
- Up to 28 analog inputs.
- Output formatted in 10- or 8-bit right-justified format.
- Single or continuous conversion (automatic return to idle after single conversion).
- Configurable sample time and conversion speed/power.
- Conversion complete flag and interrupt.
- Input clock selectable from up to four sources.
- Operation in wait or stop3 modes for lower noise operation.
- Asynchronous clock source for lower noise operation.
- Selectable asynchronous hardware conversion trigger.
- Automatic compare with interrupt for less-than, or greater-than or equal-to, programmable value.

9.1.3 Block Diagram

Figure 9-2 provides a block diagram of the ADC module

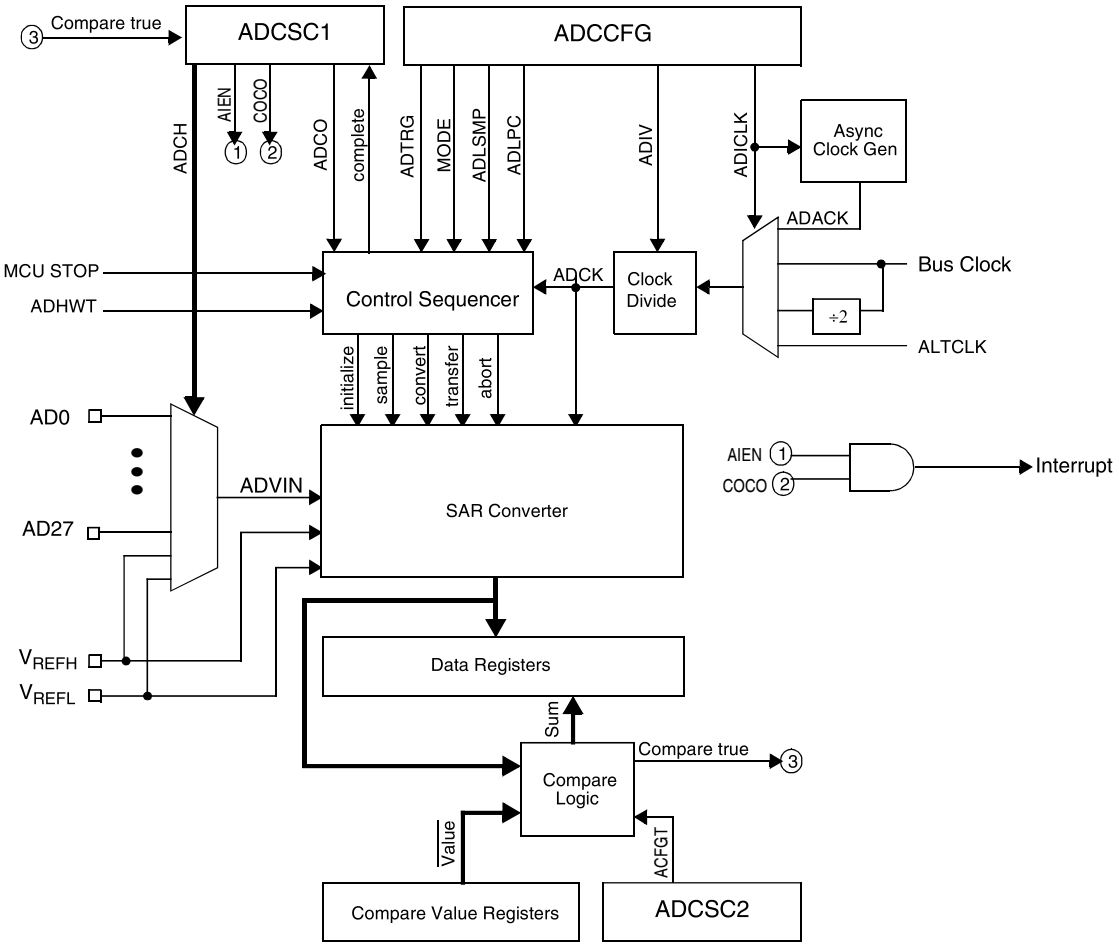


Figure 9-2. ADC Block Diagram

9.2 External Signal Description

The ADC module supports up to 28 separate analog inputs. It also requires four supply/reference/ground connections.

Table 9-2. Signal Properties

Name	Function
AD27–AD0	Analog Channel inputs
V_{REFH}	High reference voltage
V_{REFL}	Low reference voltage
V_{DDAD}	Analog power supply
V_{SSAD}	Analog ground

9.2.1 Analog Power (V_{DDAD})

The ADC analog portion uses V_{DDAD} as its power connection. In some packages, V_{DDAD} is connected internally to V_{DD}. If externally available, connect the V_{DDAD} pin to the same voltage potential as V_{DD}. External filtering may be necessary to ensure clean V_{DDAD} for good results.

9.2.2 Analog Ground (V_{SSAD})

The ADC analog portion uses V_{SSAD} as its ground connection. In some packages, V_{SSAD} is connected internally to V_{SS}. If externally available, connect the V_{SSAD} pin to the same voltage potential as V_{SS}.

9.2.3 Voltage Reference High (V_{REFH})

V_{REFH} is the high reference voltage for the converter. In some packages, V_{REFH} is connected internally to V_{DDAD}. If externally available, V_{REFH} may be connected to the same potential as V_{DDAD}, or may be driven by an external source that is between the minimum V_{DDAD} spec and the V_{DDAD} potential (V_{REFH} must never exceed V_{DDAD}).

9.2.4 Voltage Reference Low (V_{REFL})

V_{REFL} is the low reference voltage for the converter. In some packages, V_{REFL} is connected internally to V_{SSAD}. If externally available, connect the V_{REFL} pin to the same voltage potential as V_{SSAD}.

9.2.5 Analog Channel Inputs (ADx)

The ADC module supports up to 28 separate analog inputs. An input is selected for conversion through the ADCH channel select bits.

9.3 Register Definition

These memory mapped registers control and monitor operation of the ADC:

- Status and control register, ADCSC1
- Status and control register, ADCSC2
- Data result registers, ADCRH and ADCRL
- Compare value registers, ADCCVH and ADCCVL
- Configuration register, ADCCFG
- Pin enable registers, APCTL1, APCTL2, APCTL3

9.3.1 Status and Control Register 1 (ADCSC1)

This section describes the function of the ADC status and control register (ADCSC1). Writing ADCSC1 aborts the current conversion and initiates a new conversion (if the ADCH bits are equal to a value other than all 1s).

Chapter 10
Internal Clock Source (S08ICSV1)

10.1 Introduction

The internal clock source (ICS) module provides clock source choices for the MCU. The module contains a frequency-locked loop (FLL) as a clock source that is controllable by either an internal or an external reference clock. The module can provide this FLL clock or either of the internal or external reference clocks as a source for the MCU system clock. There are also signals provided to control a low power oscillator (XOSC) module to allow the use of an external crystal/resonator as the external reference clock.

Whichever clock source is chosen, it is passed through a reduced bus divider (BDIV) which allows a lower final output clock frequency to be derived.

The bus frequency will be one-half of the ICSOUT frequency.

NOTE

The external reference clock is not available on all packages. See Table 1-1 for external clock availability for each package option.

10.1.1 Module Configuration

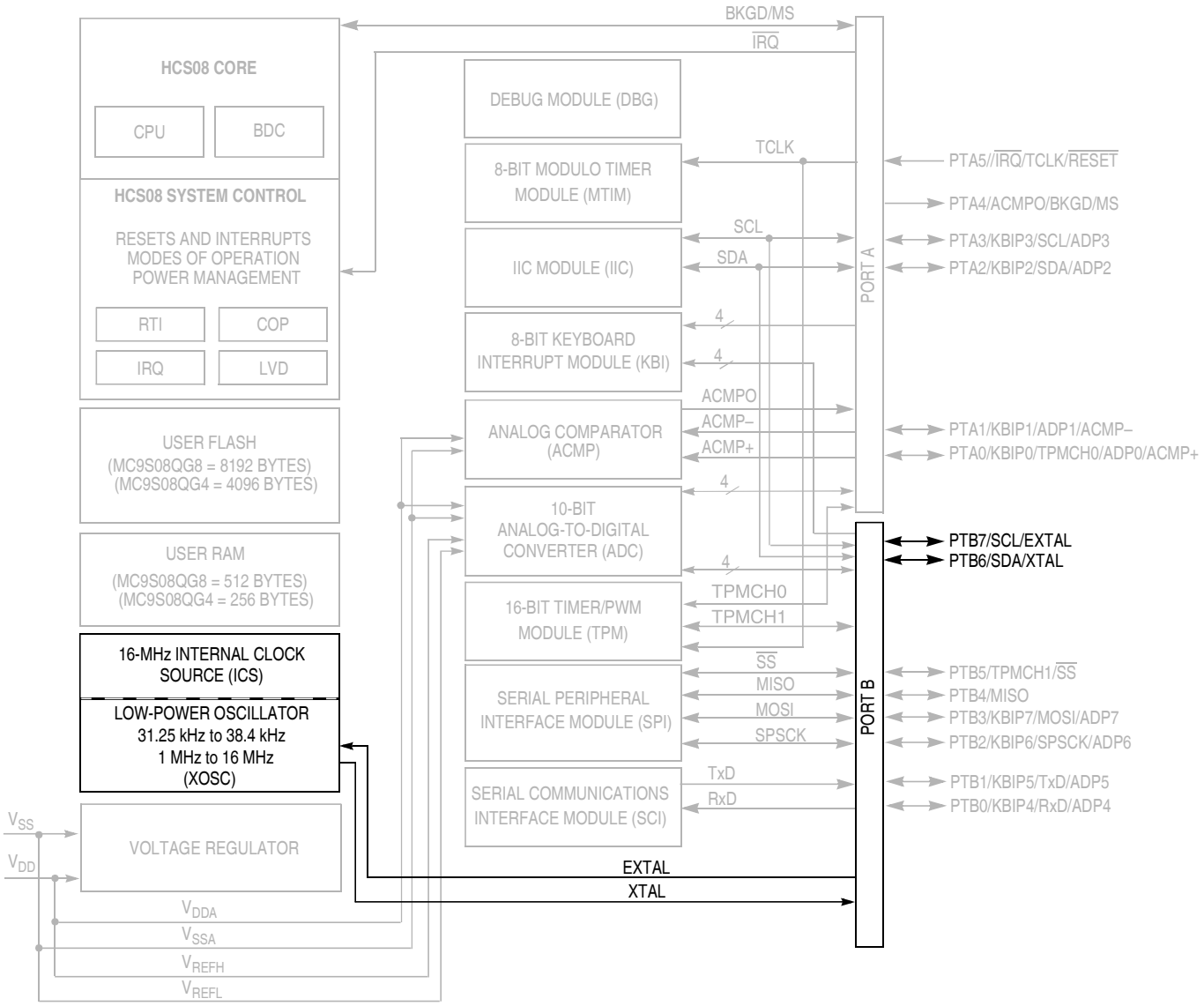
When the internal reference is enabled in stop mode (IREFSTEN = 1), the voltage regulator must also be enabled in stop mode by setting the LVDE and LVDSE bits in the SPMSC1 register.

On this MCU, the internal reference is not connected to any module that is operational in stop mode. Therefore, the IREFSTEN bit in the ICSC1 register should always be cleared.

Figure 10-1 shows the MC9S08QG8/4 block diagram with the ICS highlighted.

10.1.2 Factory Trim Value

A factory trim value is stored in FLASH during production testing. To be used, this value must be copied from FLASH memory to the ICSTRM register. A factory value for this FTRIM bit is also stored in FLASH and must be copied into the FTRIM bit in the ICSSC register. See Table 4-4 for the FLASH locations of the factory ICSTRM and FTRIM values.



NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. $\overline{IRQ}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{IRQ}$ pin function (IRQPE = 1).
6. $\overline{RESET}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 10-1. MC9S08QG8/4 Block Diagram Highlighting ICS Block and Pins

10.1.3 Features

Key features of the ICS module are:

- Frequency-locked loop (FLL) is trimmable for accuracy
  - 0.2% resolution using internal 32 kHz reference
  - 2% deviation over voltage and temperature using internal 32 kHz reference
- External reference clock up to 5 MHz can be used to control the FLL
  - 3 bit select for reference divider is provided
- Internal reference clock has 9 trim bits available
- Internal or external reference clock can be selected as the clock source for the MCU
- Whichever clock is selected as the source can be divided down
  - 2 bit select for clock divider is provided
    - Allowable dividers are: 1, 2, 4, 8
    - BDC clock is provided as a constant divide by 2 of the DCO output
- Control signals for a low power oscillator as the external reference clock are provided
  - HGO, RANGE, EREFS, ERCLKEN, EREFSTEN
- FLL engaged internal mode is automatically selected out of reset

10.1.4 Modes of Operation

The ICS features the following modes of operation: FEI, FEE, FBI, FBILP, FBE, FBELP, and stop.

10.1.4.1 FLL Engaged Internal (FEI)

In FLL engaged internal mode, which is the default mode, the ICS supplies a clock derived from the FLL which is controlled by the internal reference clock. The BDC clock is supplied from the FLL.

10.1.4.2 FLL Engaged External (FEE)

In FLL engaged external mode, the ICS supplies a clock derived from the FLL which is controlled by an external reference clock. The BDC clock is supplied from the FLL.

10.1.4.3 FLL Bypassed Internal (FBI)

In FLL bypassed internal mode, the FLL is enabled and controlled by the internal reference clock, but is bypassed. The ICS supplies a clock derived from the internal reference clock. The BDC clock is supplied from the FLL.

10.1.4.4 FLL Bypassed Internal Low Power (FBILP)

In FLL bypassed internal low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the internal reference clock. The BDC clock is not available.

10.1.4.5 FLL Bypassed External (FBE)

In FLL bypassed external mode, the FLL is enabled and controlled by an external reference clock, but is bypassed. The ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source. The BDC clock is supplied from the FLL.

10.1.4.6 FLL Bypassed External Low Power (FBELP)

In FLL bypassed external low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source. The BDC clock is not available.

10.1.4.7 Stop (STOP)

In stop mode, the FLL is disabled and the internal or external reference clock can be selected to be enabled or disabled. The BDC clock is not available. ICS does not provide an MCU clock source.

10.1.5 Block Diagram

This section contains the ICS block diagram.

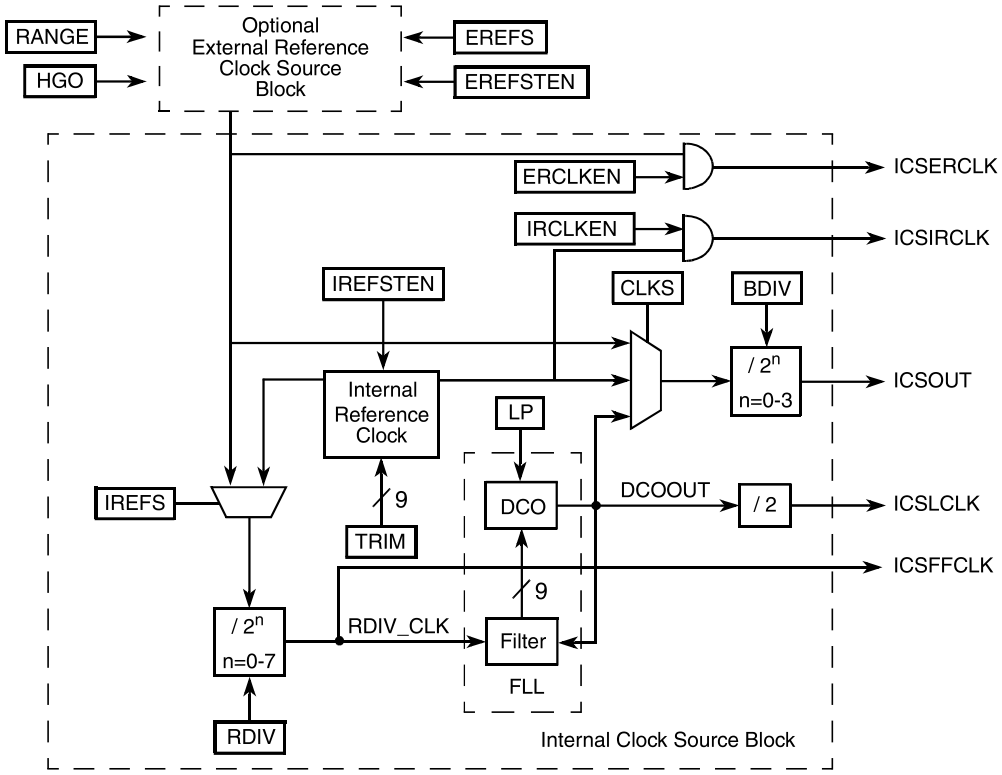


Figure 10-2. Internal Clock Source (ICS) Block Diagram

Chapter 11
Inter-Integrated Circuit (S08IICV1)

11.1 Introduction

The inter-integrated circuit (IIC) provides a method of communication between a number of devices. The interface is designed to operate up to 100 kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

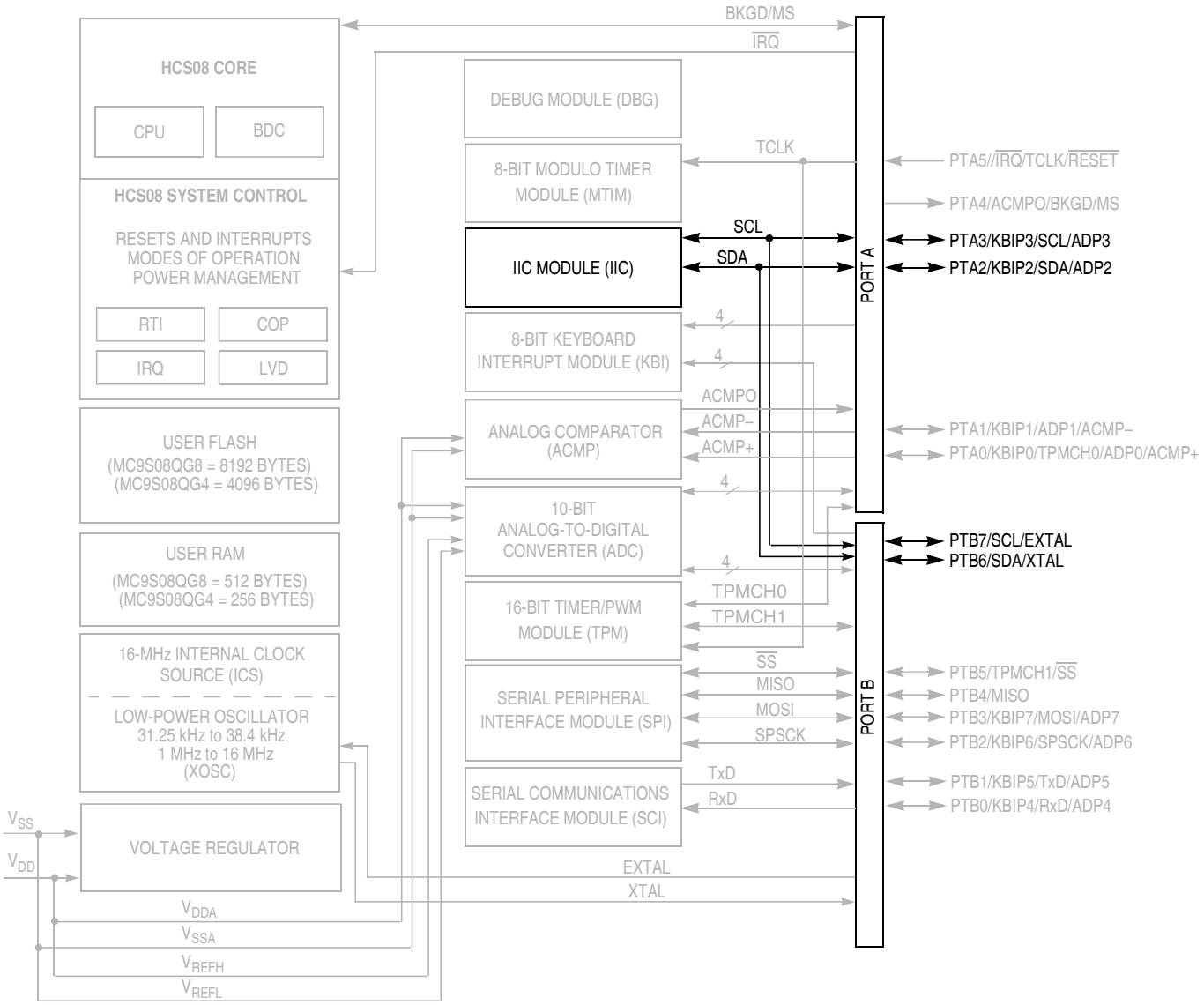
11.1.1 Module Configuration

The IIC module pins, SDA and SCL can be repositioned under software control using IICPS in SOPT2 as as shown in Table 11-1. IICPS in SOPT2 selects which general-purpose I/O ports are associated with IIC operation.

Table 11-1. IIC Position Options

IICPS in SOPT2	Port Pin for SDA	Port Pin for SCL
0 (default)	PTA2	PTA3
1	PTB6	PTB7

Figure 11-1 is the MC9S08QG8/4 block diagram with the IIC block highlighted.



NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. $\overline{IRQ}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{IRQ}$ pin function (IRQPE = 1).
6. $\overline{RESET}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 11-1. MC9S08QG8/4 Block Diagram Highlighting IIC Block and Pins

11.1.2 Features

The IIC includes these distinctive features:

- Compatible with IIC bus standard
- Multi-master operation
- Software programmable for one of 64 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Repeated START signal generation
- Acknowledge bit generation/detection
- Bus busy detection

11.1.3 Modes of Operation

The IIC functions the same in normal and monitor modes. A brief description of the IIC in the various MCU modes is given here.

- Run mode — This is the basic mode of operation. To conserve power in this mode, disable the module.
- Wait mode — The module will continue to operate while the MCU is in wait mode and can provide a wake-up interrupt.
- Stop mode — The IIC is inactive in stop3 mode for reduced power consumption. The STOP instruction does not affect IIC register states. Stop2 will reset the register contents.

11.1.4 Block Diagram

Figure 11-2 is a block diagram of the IIC.

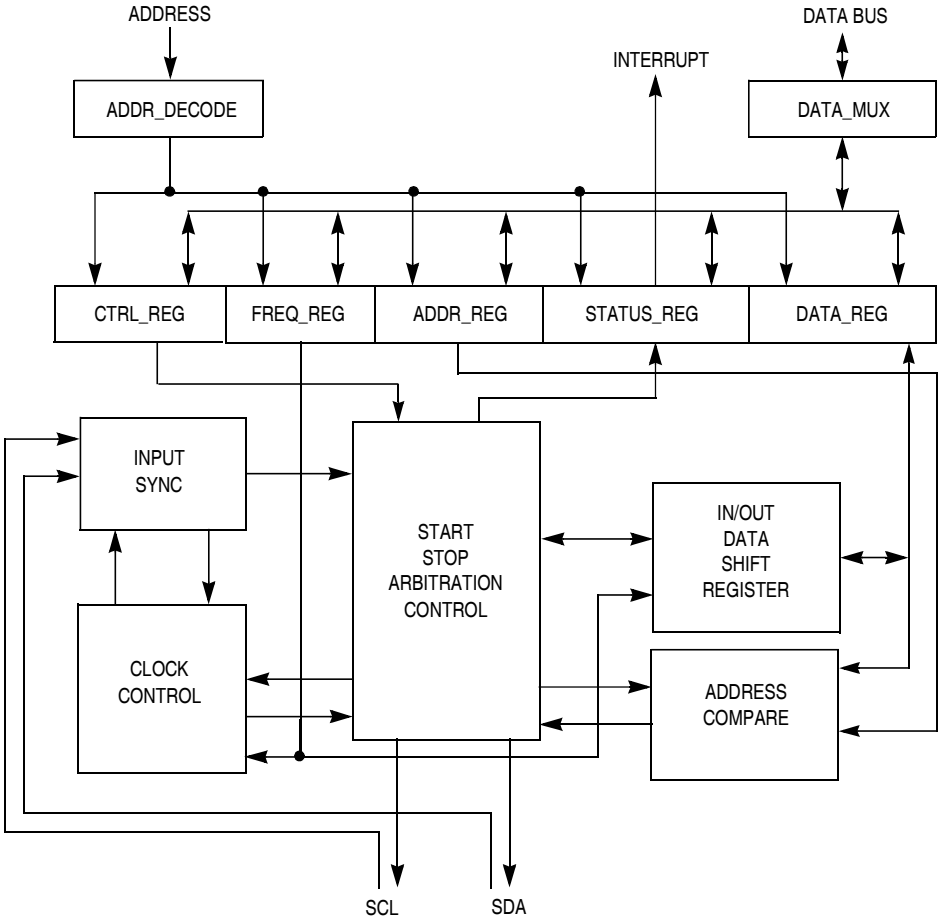


Figure 11-2. IIC Functional Block Diagram

11.2 External Signal Description

This section describes each user-accessible pin signal.

11.2.1 SCL — Serial Clock Line

The bidirectional SCL is the serial clock line of the IIC system.

11.2.2 SDA — Serial Data Line

The bidirectional SDA is the serial data line of the IIC system.

11.3 Register Definition

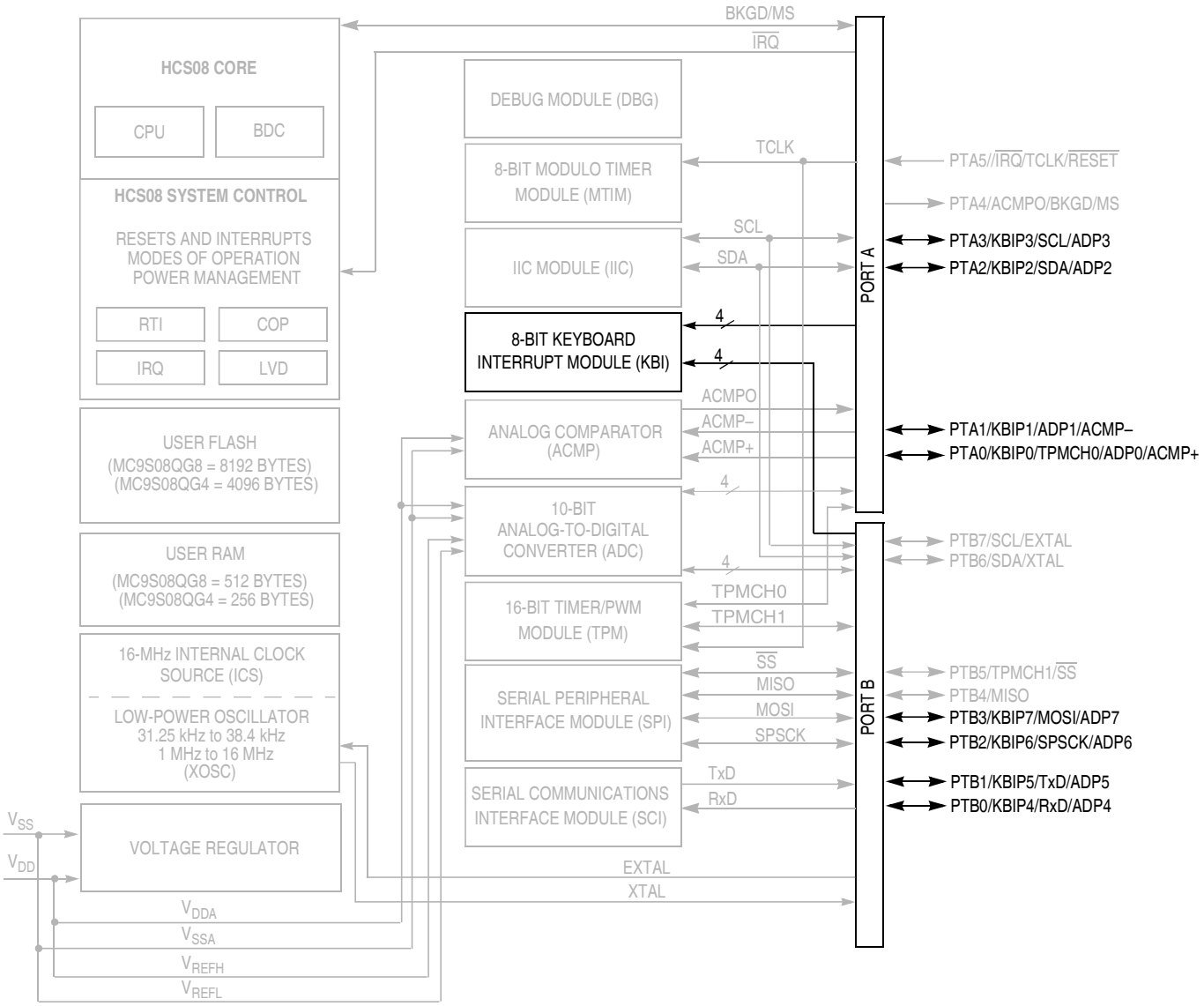
This section consists of the IIC register descriptions in address order.

Chapter 12
Keyboard Interrupt (S08KBIV2)

12.1 Introduction

The keyboard interrupt KBI module provides up to eight independently enabled external interrupt sources.

Figure 12-1 Shows the MC9S08QG8/4 block guide with the KBI highlighted.



NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. $\overline{IRQ}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{IRQ}$ pin function (IRQPE = 1).
6. $\overline{RESET}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 12-1. MC9S08QG8/4 Block Diagram Highlighting KBI Block and Pins

12.1.1 Features

The KBI features include:

- Up to eight keyboard interrupt pins with individual pin enable bits.
- Each keyboard interrupt pin is programmable as falling edge (or rising edge) only, or both falling edge and low level (or both rising edge and high level) interrupt sensitivity.
- One software enabled keyboard interrupt.
- Exit from low-power modes.

12.1.2 Modes of Operation

This section defines the KBI operation in wait, stop, and background debug modes.

12.1.2.1 KBI in Wait Mode

The KBI continues to operate in wait mode if enabled before executing the WAIT instruction. Therefore, an enabled KBI pin (KBPEx = 1) can be used to bring the MCU out of wait mode if the KBI interrupt is enabled (KBIE = 1).

12.1.2.2 KBI in Stop Modes

The KBI operates asynchronously in stop3 mode if enabled before executing the STOP instruction. Therefore, an enabled KBI pin (KBPEx = 1) can be used to bring the MCU out of stop3 mode if the KBI interrupt is enabled (KBIE = 1).

During either stop1 or stop2 mode, the KBI is disabled. In some systems, the pins associated with the KBI may be sources of wakeup from stop1 or stop2, see the stop modes section in the Modes of Operation chapter. Upon wake-up from stop1 or stop2 mode, the KBI module will be in the reset state.

12.1.2.3 KBI in Active Background Mode

When the microcontroller is in active background mode, the KBI will continue to operate normally.

12.1.3 Block Diagram

The block diagram for the keyboard interrupt module is shown Figure 12-2.

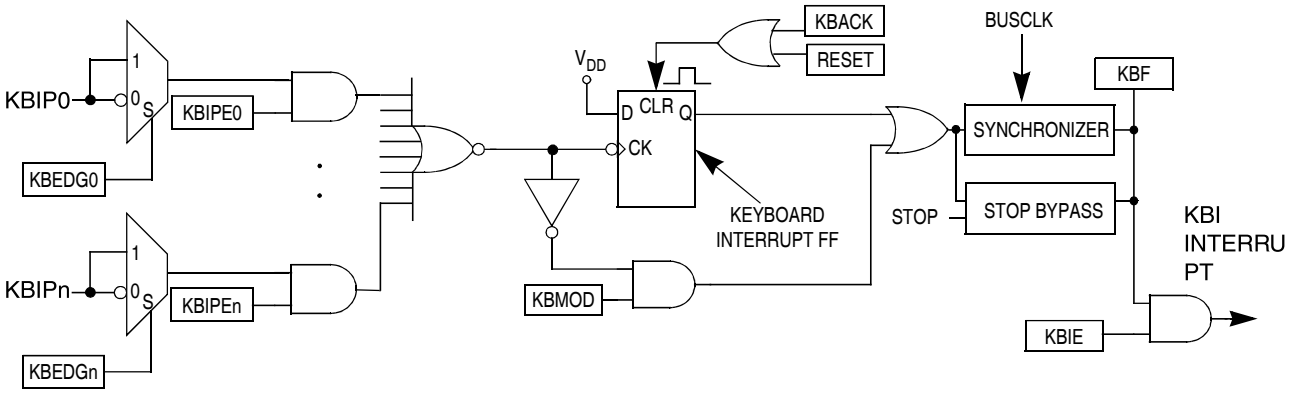


Figure 12-2. KBI Block Diagram

12.2 External Signal Description

The KBI input pins can be used to detect either falling edges, or both falling edge and low level interrupt requests. The KBI input pins can also be used to detect either rising edges, or both rising edge and high level interrupt requests.

The signal properties of KBI are shown in Table 12-1.

Table 12-1. Signal Properties

Signal	Function	I/O
KBIPn	Keyboard interrupt pins	I

12.3 Register Definition

The KBI includes three registers:

- An 8-bit pin status and control register.
- An 8-bit pin enable register.
- An 8-bit edge select register.

Refer to the direct-page register summary in the Memory chapter for the absolute address assignments for all KBI registers. This section refers to registers and control bits only by their names.

Some MCUs may have more than one KBI, so register names include placeholder characters to identify which KBI is being referenced.

12.3.1 KBI Status and Control Register (KBISC)

KBISC contains the status flag and control bits, which are used to configure the KBI.

Chapter 13
Modulo Timer (S08MTIMV1)

13.1 Introduction

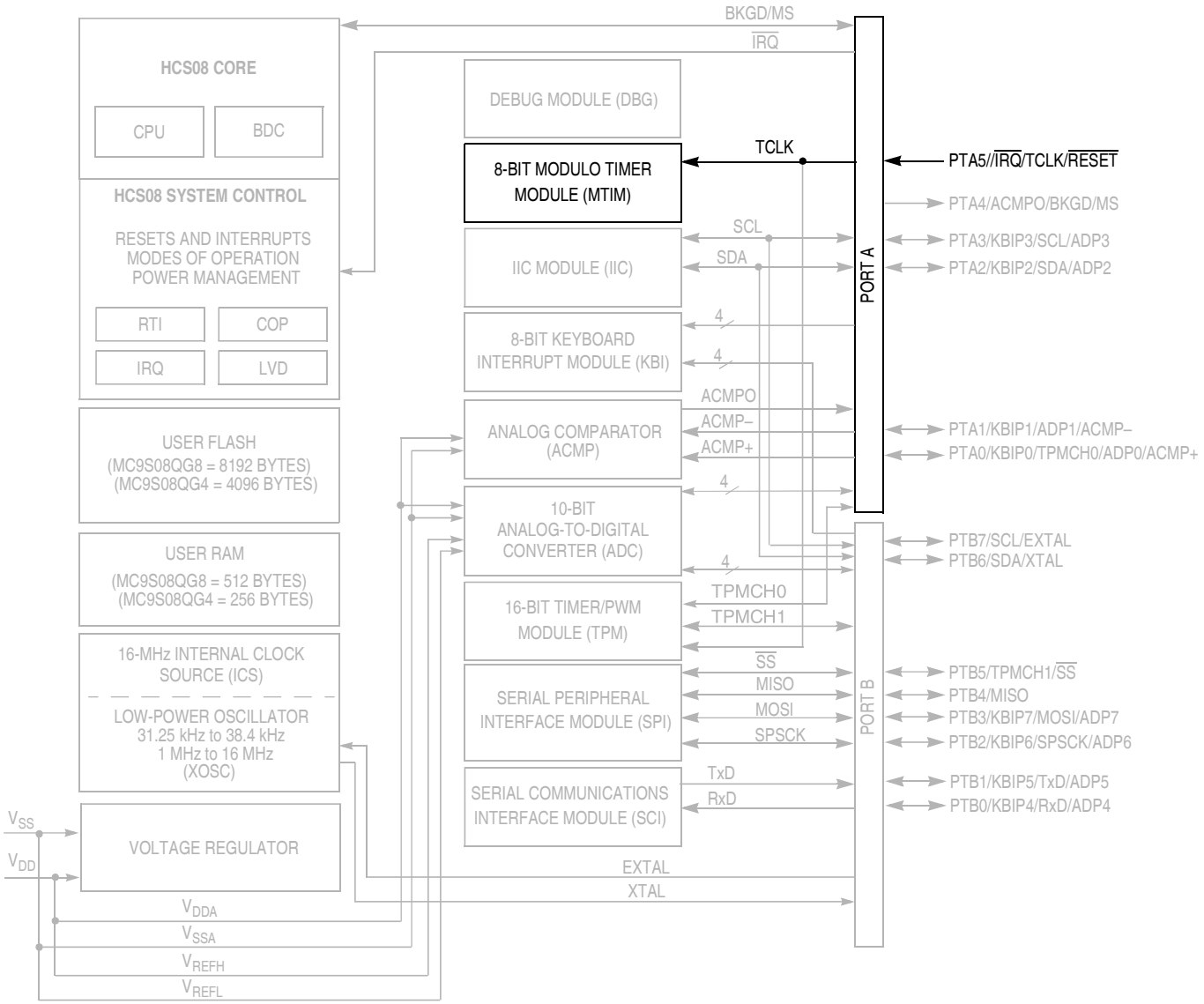
The MTIM is a simple 8-bit timer with several software selectable clock sources and a programmable interrupt.

The central component of the MTIM is the 8-bit counter, which can operate as a free-running counter or a modulo counter. A timer overflow interrupt can be enabled to generate periodic interrupts for time-based software loops.

Figure 13-1 shows the MC9S08QG8/4 block diagram with the MTIM highlighted.

13.1.1 MTIM/TPM Configuration Information

The external clock for the MTIM module, TCLK, is selected by setting CLKS = 1:1 or 1:0 in MTIMCLK, which selects the TCLK pin input. The TCLK input on PTA5 can be enabled as external clock inputs to both the MTIM and TPM modules simultaneously.



NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. $\overline{IRQ}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{IRQ}$ pin function (IRQPE = 1).
6. $\overline{RESET}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 13-1. MC9S08QG8/4 Block Diagram Highlighting MTIM Block and Pins

13.1.2 Features

Timer system features include:

- 8-bit up-counter
  - — Free-running or 8-bit modulo limit
  - — Software controllable interrupt on overflow
  - — Counter reset bit (TRST)
  - — Counter stop bit (TSTP)
- Four software selectable clock sources for input to prescaler:
  - — System bus clock — rising edge
  - — Fixed frequency clock (XCLK) — rising edge
  - — External clock source on the TCLK pin — rising edge
  - — External clock source on the TCLK pin — falling edge
- Nine selectable clock prescale values:
  - — Clock source divide by 1, 2, 4, 8, 16, 32, 64, 128, or 256

13.1.3 Modes of Operation

This section defines the MTIM's operation in stop, wait and background debug modes.

13.1.3.1 MTIM in Wait Mode

The MTIM continues to run in wait mode if enabled before executing the WAIT instruction. Therefore, the MTIM can be used to bring the MCU out of wait mode if the timer overflow interrupt is enabled. For lowest possible current consumption, the MTIM should be stopped by software if not needed as an interrupt source during wait mode.

13.1.3.2 MTIM in Stop Modes

The MTIM is disabled in all stop modes, regardless of the settings before executing the STOP instruction. Therefore, the MTIM cannot be used as a wake up source from stop modes.

Waking from stop1 and stop2 modes, the MTIM will be put into its reset state. If stop3 is exited with a reset, the MTIM will be put into its reset state. If stop3 is exited with an interrupt, the MTIM continues from the state it was in when stop3 was entered. If the counter was active upon entering stop3, the count will resume from the current value.

13.1.3.3 MTIM in Active Background Mode

The MTIM suspends all counting until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as an MTIM reset did not occur (TRST written to a 1 or MTIMMOD written).

13.1.4 Block Diagram

The block diagram for the modulo timer module is shown Figure 13-2.

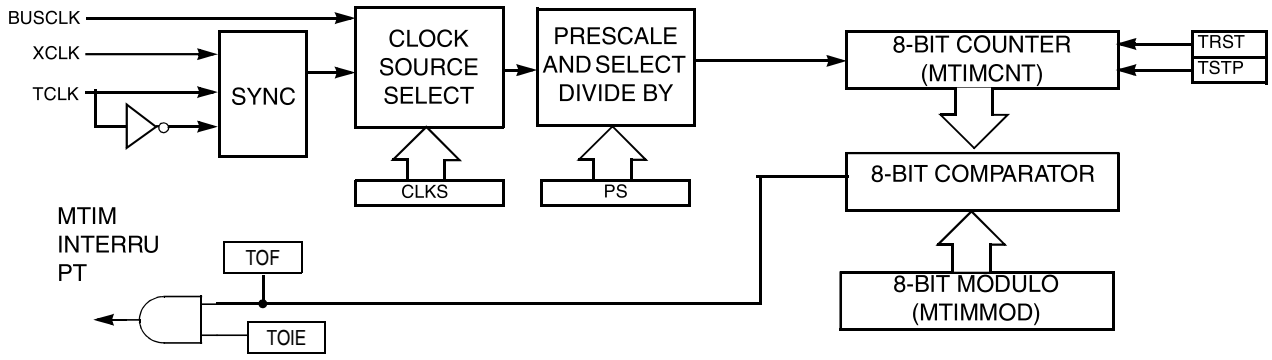


Figure 13-2. Modulo Timer (MTIM) Block Diagram

13.2 External Signal Description

The MTIM includes one external signal, TCLK, used to input an external clock when selected as the MTIM clock source. The signal properties of TCLK are shown in Table 13-1.

Table 13-1. Signal Properties

Signal	Function	I/O
TCLK	External clock source input into MTIM	I

The TCLK input must be synchronized by the bus clock. Also, variations in duty cycle and clock jitter must be accommodated. Therefore, the TCLK signal must be limited to one-fourth of the bus frequency.

The TCLK pin can be muxed with a general-purpose port pin. See the Pins and Connections chapter for the pin location and priority of this function.

13.3 Register Definition

Figure 13-3 is a summary of MTIM registers.

Figure 13-3. MTIM **Register Summary**

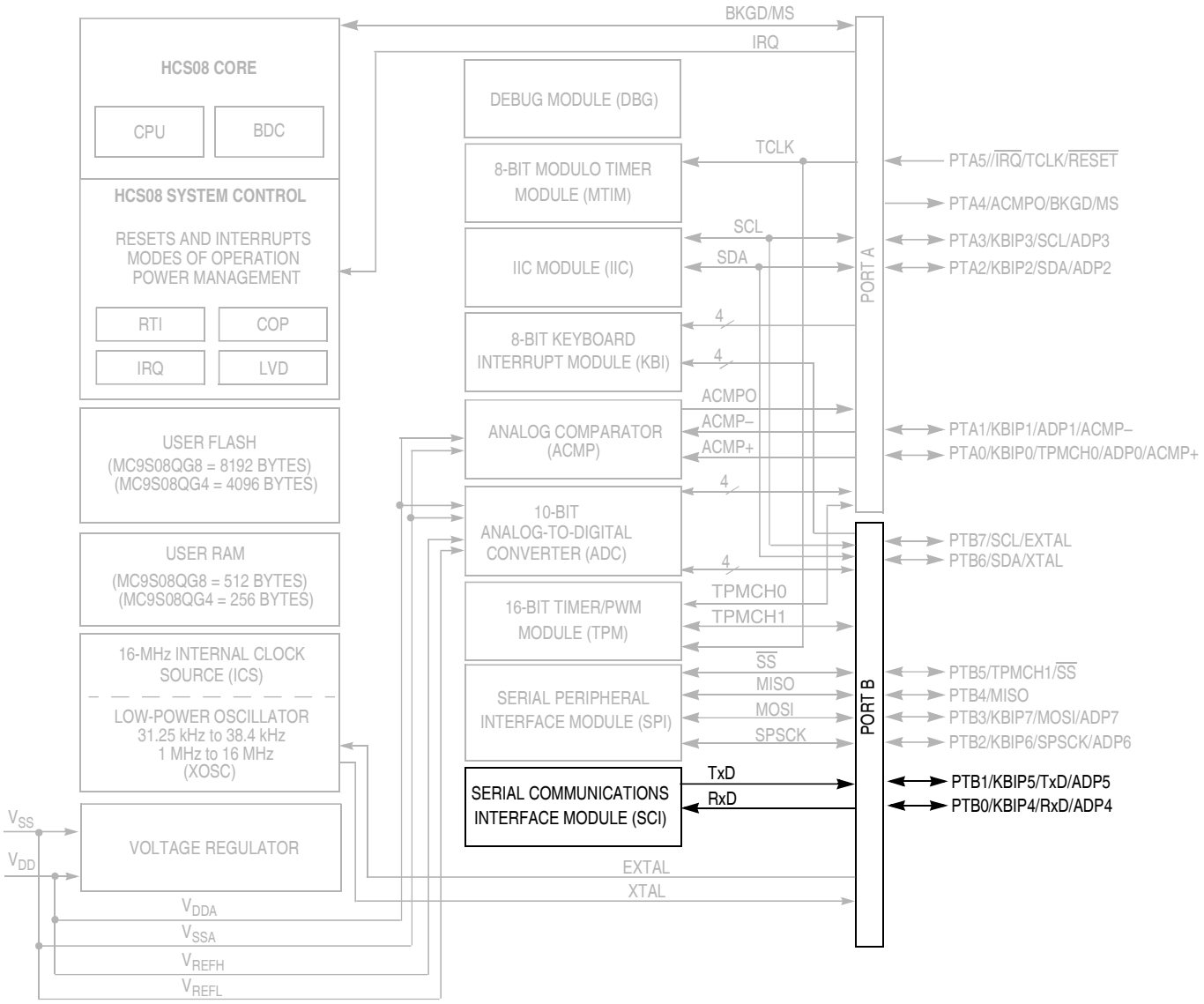
Name		7	6	5	4	3	2	1	0
MTIMSC	R	TOF	TOIE	0	TSTP	0	0	0	0
	W			TRST					
MTIMCLK	R	0	0	CLKS		PS			
	W								
MTIMCNT	R	COUNT							
	W								
MTIMMOD	R	MOD							
	W								

Each MTIM includes four registers:

- An 8-bit status and control register
- An 8-bit clock configuration register
- An 8-bit counter register
- An 8-bit modulo register

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all MTIM registers. This section refers to registers and control bits only by their names and relative address offsets.

Some MCUs may have more than one MTIM, so register names include placeholder characters to identify which MTIM is being referenced.



NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. IRQ contains a software configurable (IRQPDD) pullup/pulldown device if PTA5 enabled as IRQ pin function (IRQPE = 1).
6. $\overline{\text{RESET}}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 14-1. MC9S08QG8/4 Block Diagram Highlighting SCI Block and Pins

14.1.1 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character
- Selectable transmitter output polarity

14.1.2 Modes of Operation

See Section 14.3, “Functional Description,” for a detailed description of SCI operation in the different modes.

- 8- and 9- bit data modes
- Stop modes — SCI is halted during all stop modes
- Loop mode
- Single-wire mode

Anexo III. Datasheet ESP8266

1. Preambles

ESP-01 WiFi module is developed by Ai-thinker Team. core processor ESP8266 in smaller sizes of the module encapsulates Tensilica L106 integrates industry-leading ultra low power 32-bit MCU micro, with the 16-bit short mode, Clock speed support 80 MHz, 160 MHz, supports the RTOS, integrated Wi-Fi MAC/BB/RF/PA/LNA, on-board antenna.

The module supports standard IEEE802.11 b/g/n agreement, complete TCP/IP protocol stack. Users can use the add modules to an existing device networking, or building a separate network controller.

ESP8266 is high integration wireless SOCs, designed for space and power constrained mobile platform designers. It provides unsurpassed ability to embed Wi-Fi capabilities within other systems, or to function as a standalone application, with the lowest cost, and minimal space requirement.

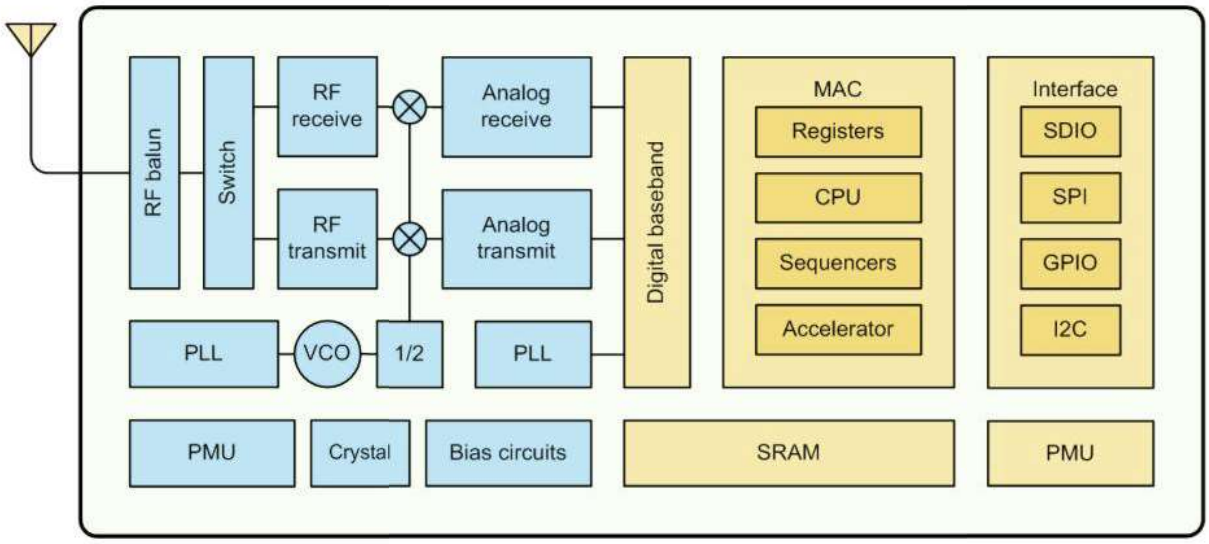


Figure 1 ESP8266EX Block Diagram

ESP8266EX offers a complete and self-contained Wi-Fi networking solution; it can be used to host the application or to offload Wi-Fi networking functions from another application processor.

When ESP8266EX hosts the application, it boots up directly from an external flash. In has integrated cache to improve the performance of the system in such applications.

Alternately, serving as a Wi-Fi adapter, wireless internet access can be added to any micro controllerbased design with simple connectivity (SPI/SDIO or I2C/UART interface).

1.2. Parameters

Table 1 below describes the major parameters.

Table 1 Parameters

Categories	Items	Values
WiFi Paramters	WiFi Protocles	802.11 b/g/n
	Frequency Range	2.4GHz-2.5GHz (2400M-2483.5M)
Hardware Paramaters	Peripheral Bus	UART/HSPI/I2C/I2S/Ir Remote Contorl
		GPIO/PWM
	Operating Voltage	3.0~3.6V
	Operating Current	Average value: 80mA
	Operating Temperature Range	-40°~125°
	Ambient Temperature Range	Normal temperature
	Package Size	14.3mm*24.8mm*3mm
	External Interface	N/A
Software Parameters	Wi-Fi mode	station/softAP/SoftAP+station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network) / download and write firmware via host
	Ssoftware Development	Supports Cloud Server Development / SDK for custom firmware development
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

2. Pin Descriptions

There are altogether 8 pin counts, the definitions of which are described in Table 2 below。

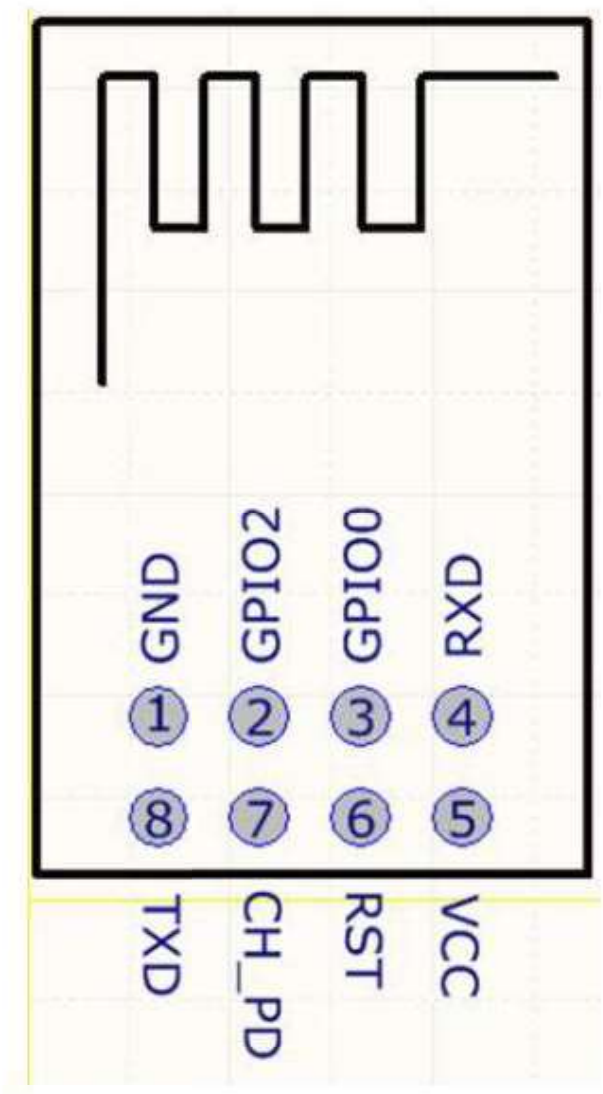


Table 2 ESP-01 Pin design

Table 2 Pin Descriptions

NO.	Pin Name	Function
1	GND	GND
2	GPIO2	GPIO,Internal Pull-up
3	GPIO0	GPIO,Internal Pull-up
4	RXD	UART0,data received pin RXD
5	VCC	3.3V power supply (VDD)
6	RST	1) External reset pin, active low 2) Can loft or external MCU ,
7	CH_PD	Chip enable pin. Active high
8	TXD	UART0,data send pin RXD

Table 3 Pin Mode

Mode	GPIO15	GPIO0	GPIO2
UART	Low	Low	High
Flash Boot	Low	High	High

Table 4 Receiver Sensitivity

Parameters	Min	Typical	Max	Unit
Input frequency	2412		2484	MHz
Input impedance		50		Ω
Input reflection			-10	dB
Output power of PA for 72.2Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
Sensitivity				
DSSS, 1Mbps		-98		dBm
CCK, 11Mbps		-91		dBm
6Mbps (1/2 BPSK)		-93		dBm
54Mbps (3/4 64-QAM)		-75		dBm
HT20, MCS7 (65Mbps, 72.2Mbps)		-72		dBm
Adjacent Channel Rejection				
OFDM, 6Mbps		37		dB
OFDM, 54Mbps		21		dB
HT20, MCS0		37		dB
HT20, MCS7		20		dB

3. Packaging and Dimension

The external size of the module is 14.3mm*24.8mm*3mm, as is illustrated in Figure 3 below. The type of flash integrated in this module is an SPI flash, the capacity of which is 1 MB, and the package size of which is SOP-210mil. The antenna applied on this module is a 3DBi PCB-on-board antenna.

Figure 3 [Module Pin Counts, 8 pin, 14.3 mm *24.8 mm *3.0 mm]

4.3. Crystal

Currently, the frequency of crystal oscillators supported include 40MHz, 26MHz and 24MHz. The accuracy of crystal oscillators applied should be ±10PPM, and the operating temperature range should be between -20°C and 85°C.

When using the downloading tools, please remember to select the right crystal oscillator type. In circuit design, capacitors C1 and C2, which are connected to the earth, are added to the input and output terminals of the crystal oscillator respectively. The values of the two capacitors can be flexible, ranging from 6pF to 22pF, however, the specific capacitive values of C1 and C2 depend on further testing and adjustment on the overall performance of the whole circuit. Normally, the capacitive values of C1 and C2 are within 10pF if the crystal oscillator frequency is 26MHz, while the values of C1 and C2 are 10pF<C1, C2<22pF if the crystal oscillator frequency is 40MHz.

4.4. Interfaces

Table 6 Descriptions of Interfaces

Interface	Pin Name	Description
HSPI	IO12(MISO) IO13(MOSI) IO14(CLK) IO15(CS)	SPI Flash 2, display screen, and MCU can be connected using HSPI interface.
PWM	IO12(R) IO15(G) IO13(B)	Currently the PWM interface has four channels, but users can extend the channels according to their own needs. PWM interface can be used to control LED lights, buzzers, relays, electronic machines, and so on.
IR Remote Control	IO14(IR_T) IO5(IR_R)	The functionality of Infrared remote control interface can be implemented via software programming. NEC coding, modulation, and demodulation are used by this interface. The frequency of modulated carrier signal is 38KHz.
ADC	TOUT	ESP8266EX integrates a 10-bit analog ADC. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously. This interface is typically used in sensor products.
I2C	IO14(SCL) IO2(SDA)	I2C interface can be used to connect external sensor products and display screens, etc.

Interface	Pin Name	Description
UART	**UART0:** TXD (U0TXD) RXD (U0RXD) IO15 (RTS) IO13 (CTS) **UART1:** IO2(TXD)	Devices with UART interfaces can be connected with the module. Downloading: U0TXD+U0RXD or GPIO2+U0RXD Communicating: UART0: U0TXD, U0RXD, MTDO (U0RTS), MTCK (U0CTS) Debugging: UART1_TXD (GPIO2) can be used to print debugging information. By default, UART0 will output some printed information when the device is powered on and is booting up. If this issue exerts influence on some specific applications, users can exchange the inner pins of UART when initializing, that is to say, exchange U0TXD, U0RXD with U0RTS, U0CTS.
I2S	**I2S Input:** IO12 (I2SI_DATA) ; IO13 (I2SI_BCK); IO14 (I2SI_WS); **I2S Output:** IO15 (I2SO_BCK); IO3 (I2SO_DATA); IO2 (I2SO_WS).	I2S interface is mainly used for collecting, processing, and transmission of audio data.

5. RF Performance

Description	Min.	Typ.	Max	Unit
Input frequency	2400		2483.5	MHz
Input impedance		50		ohm
Input reflection			-10	dB
Output power of PA for 72.2Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
Sensitivity				
CCK, 1Mbps		-98		dBm
CCK, 11Mbps		-91		dBm
6Mbps (1/2 BPSK)		-93		dBm
54Mbps (3/4 64-QAM)		-75		dBm
HT20, MCS7 (65Mbps, 72.2Mbps)		-72		dBm
Adjacent Channel Rejection				
OFDM, 6Mbps		37		dB
OFDM, 54Mbps		21		dB
HT20, MCS0		37		dB
HT20, MCS7		20		dB

Table 10 RF Performance

6. Power Consumption

Parameters	Min	Typical	Max	Unit
Tx802.11b, CCK 11Mbps, P OUT=+17dBm		170		mA
Tx 802.11g, OFDM 54Mbps, P OUT =+15dBm		140		mA
Tx 802.11n, MCS7, P OUT =+13dBm		120		mA
Rx 802.11b, 1024 bytes packet length , -80dBm		50		mA
Rx 802.11g, 1024 bytes packet length, -70dBm		56		mA
Rx 802.11n, 1024 bytes packet length, -65dBm		56		mA
Modem-Sleep①		15		mA
Light-Sleep②		0.9		mA
Deep-Sleep③		10		uA

Table 11 Power Consumption

❶Modem-Sleep requires the CPU to be working, as in PWM or I2S applications. According to 802.11 standards (like U-APSD), it saves power to shut down the Wi-Fi Modem circuit while maintaining a Wi-Fi connection with no data transmission. E.g. in DTIM3, to maintain a sleep 300mswake 3ms cycle to receive AP's Beacon packages, the current is about 15mA.

❷ During Light-Sleep, the CPU may be suspended in applications like Wi-Fi switch. Without data transmission, the Wi-Fi Modem circuit can be turned off and CPU suspended to save power according to the 802.11 standard (U-APSD). E.g. in DTIM3, to maintain a sleep 300ms-wake 3ms cycle to receive AP's Beacon packages, the current is about 0.9mA.

❸ Deep-Sleep does not require Wi-Fi connection to be maintained. For application with long time lags between data transmission, e.g. a temperature sensor that checks the temperature every 100s ,sleep 300s and waking up to connect to the AP (taking about 0.3~1s), the overall average current is less than 1mA.

8. Schematics

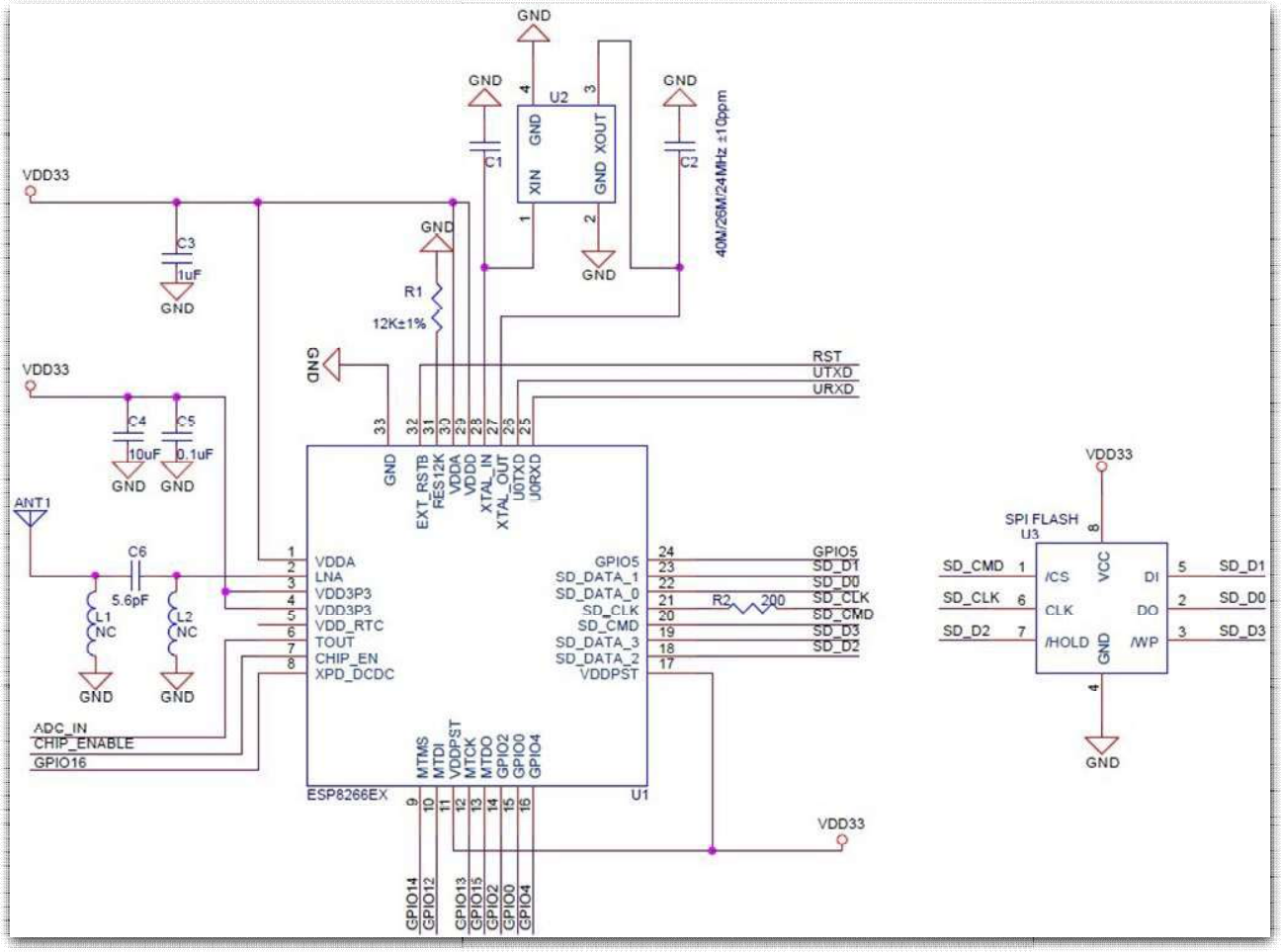
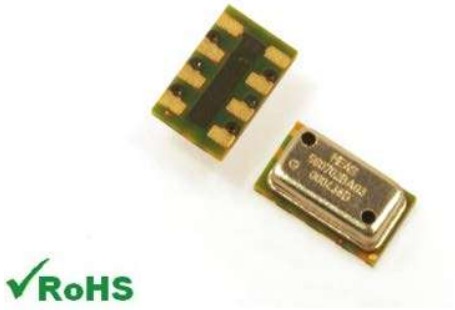


Figure 4 Schematics of Esp-01 WiFi Module

Anexo IV. Datasheet Sensor de Presión

MS5607-02BA03 Barometric Pressure Sensor, with stainless steel cap



- **High resolution module, 20cm**
- **Fast conversion down to 1 ms**
- **Low power, 1 µA (standby < 0.15 µA)**
- **QFN package 5.0 x 3.0 x 1.0 mm^3**
- **Supply voltage 1.8 to 3.6 V**
- **Integrated digital pressure sensor (24 bit ΔΣ ADC)**
- **Operating range: 10 to 1200 mbar, -40 to +85 °C**
- **I^{2}C and SPI interface up to 20 MHz**
- **No external components (Internal oscillator)**
- **Excellent long term stability**

DESCRIPTION

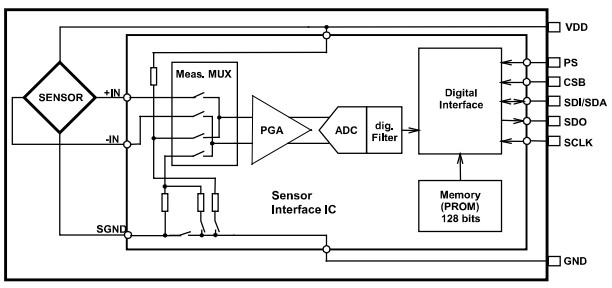
The MS5607-02BA is a new generation of high resolution altimeter sensors from MEAS Switzerland with SPI and I^{2}C bus interface. This barometric pressure sensor is optimized for altimeters and variometers with an altitude resolution of 20 cm. The sensor module includes a high linearity pressure sensor and an ultra low power 24 bit ΔΣ ADC with internal factory calibrated coefficients. It provides a precise digital 24 bit pressure and temperature value and different operation modes that allow the user to optimize for conversion speed and current consumption. A high resolution temperature output allows the implementation of an altimeter/thermometer function without any additional sensor. The MS5607-02BA can be interfaced to virtually any microcontroller. The communication protocol is simple, without the need of programming internal registers in the device. Small dimensions of only 5.0 mm x 3.0 mm and a height of only 1.0 mm allow for integration in mobile devices. This new sensor module generation is based on leading MEMS technology and latest benefits from MEAS Switzerland proven experience and know-how in high volume manufacturing of altimeter modules, which have been widely used for over a decade. The sensing principle employed leads to very low hysteresis and high stability of both pressure and temperature signal.

FEATURES

FIELD OF APPLICATION

Mobile altimeter / barometer systems
Bike computers
Variometers
Dataloggers
Mobile phones / GPS

FUNCTIONAL BLOCK DIAGRAM



TECHNICAL DATA

Sensor Performances (V_{DD} = 3 V)				
Pressure	**Min**	**Typ**	**Max**	**Unit**
Range	10		1200	mbar
ADC		24		bit
Resolution (1)		0.13 / 0.084 / 0.054 / 0.036 / 0.024		mbar
Accuracy 25°C, 750 mbar	-1.5		+1.5	mbar
Error band, -20°C to + 85°C 300 to 1100 mbar (2)	-2.5		+2.5	mbar
Response time (1)		0.5 / 1.1 / 2.1 / 4.1 / 8.22		ms
Long term stability		±1		mbar/yr
Temperature	**Min**	**Typ**	**Max**	**Unit**
Range	-40		+85	°C
Resolution		<0.01		°C
Accuracy	-0.8		+0.8	°C

Notes: (1) Oversampling Ratio: 256 / 512 / 1024 / 2048 / 4096
(2) With autozero at one pressure point

PERFORMANCE SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS

Parameter	Symbol	Conditions	Min.	Typ.	Max	Unit
Supply voltage	V_{DD}		-0.3		+4.0	V
Storage temperature	T_S		-40		+125	°C
Overpressure	P_{max}				6	bar
Maximum Soldering Temperature	T_{max}	40 sec max			250	°C
ESD rating		Human Body Model	-4		+4	kV
Latch up		JEDEC standard No 78	-100		+100	mA

ELECTRICAL CHARACTERISTICS

Parameter	Symbol	Conditions	Min.	Typ.	Max	Unit
Operating Supply voltage	V_{DD}		1.8	3.0	3.6	V
Operating Temperature	T		-40	+25	+85	°C
Supply current (1 sample per sec.)	I_{DD}	OSR 4096		12.5		µA
		2048		6.3		
		1024		3.2		
		512		1.7		
		256		0.9		
Peak supply current		during conversion		1.4		mA
Standby supply current		at 25°C		0.02	0.14	µA
VDD Capacitor		From VDD to GND	100			nF

ANALOG DIGITAL CONVERTER (ADC)

Parameter	Symbol	Conditions	Min.	Typ.	Max	Unit
Output Word				24		bit
Conversion time	t_c	OSR 4096	7.40	8.22	9.04	ms
		2048	3.72	4.13	4.54	
		1024	1.88	2.08	2.28	
		512	0.95	1.06	1.17	
		256	0.48	0.54	0.60	

PERFORMANCE SPECIFICATIONS (CONTINUED)

PRESSURE OUTPUT CHARACTERISTICS (V_{DD} = 3 V, T = 25°C UNLESS OTHERWISE NOTED)

Parameter	Conditions		Min.	Typ.	Max	Unit
Operating Pressure Range	P_{range}	Full Accuracy	300		1100	mbar
Extended Pressure Range	P_{ext}	Linear Range of ADC	10		1200	mbar
Total Error band, no autozero	at 25°C, 700..1100 mbar at 0..50°C, 300..1100 mbar at -20..85°C, 300..1100 mbar		-1.5 -2.0 -3.5		+1.5 +2.0 +3.5	mbar
Total Error band, autozero at one pressure point	at 25°C, 700..1100 mbar at 0..50°C, 300..1100 mbar at -20..85°C, 300..1100 mbar		-0.5 -1.0 -2.5		+0.5 +1.0 +2.5	mbar
Maximum error with supply voltage	V_{DD} = 1.8 V … 3.6 V		-2.5		+2.5	mbar
Resolution RMS	OSR	4096 2048 1024 512 256		0.024 0.036 0.054 0.084 0.130		mbar
Long-term stability				±1		mbar/yr
Reflow soldering impact	IPC/JEDEC J-STD-020C (See application note AN808)			+0.4		mbar
Recovering time after reflow (1)				7		days

(1) Time to recovering at least 66% of the reflow impact

TEMPERATURE OUTPUT CHARACTERISTICS (V_{DD} = 3 V, T = 25°C UNLESS OTHERWISE NOTED)

Parameter	Conditions		Min.	Typ.	Max	Unit
Absolute Accuracy	at 25°C -20..85°C -40..85°C		-0.8 -2.0 -4.0		+0.8 +2.0 +4.0	°C
Maximum error with supply voltage	V_{DD} = 1.8 V … 3.6 V		-0.5		+0.5	°C
Resolution RMS	OSR	4096 2048 1024 512 256		0.002 0.003 0.005 0.008 0.012		°C

PERFORMANCE SPECIFICATIONS (CONTINUED)

DIGITAL INPUTS (CSB, I^2C, DIN, SCLK)

Parameter	Symbol	Conditions	Min.	Typ.	Max	Unit
Serial data clock	SCLK	SPI protocol			20	MHz
		I^2C protocol			400	KHz
Input high voltage	V_{IH}	Pins CSB	80% V_{DD}		100% V_{DD}	V
Input low voltage	V_{IL}		0% V_{DD}		20% V_{DD}	V
Input leakage current	$I_{leak25°C}$ $I_{leak85°C}$	at 25°C			0.15	µA
Input capacitance	C_{IN}				6	pF

PRESSURE OUTPUTS (I^2C, DOUT)

Parameter	Symbol	Conditions	Min.	Typ.	Max	Unit
Output high voltage	V_{OH}	I_{source} = 1.0 mA	80% V_{DD}		100% V_{DD}	V
Output low voltage	V_{OL}	I_{sink} = 1.0 mA	0% V_{DD}		20% V_{DD}	V
Load capacitance	C_{LOAD}				16	pF

FUNCTIONAL DESCRIPTION

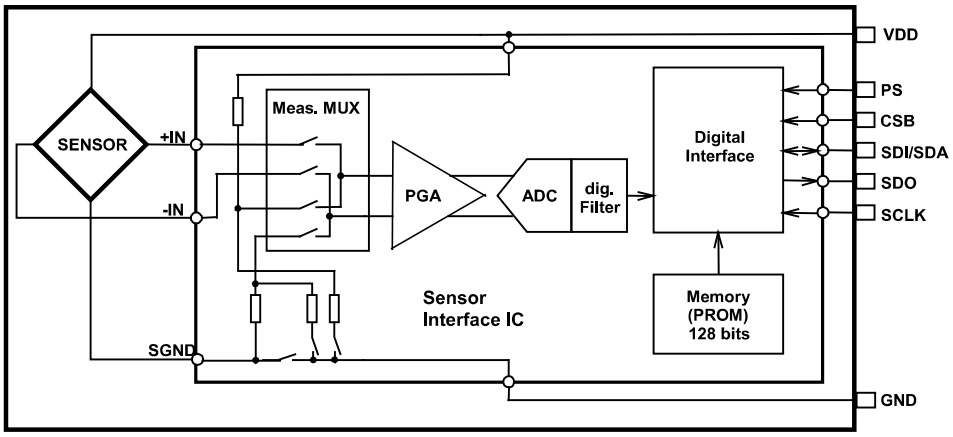


Figure 1: Block diagram of MS5607-02BA

GENERAL

The MS5607-02BA consists of a piezo-resistive sensor and a sensor interface IC. The main function of the MS5607-02BA is to convert the uncompensated analogue output voltage from the piezo-resistive pressure sensor to a 24-bit digital value, as well as providing a 24-bit digital value for the temperature of the sensor.

FACTORY CALIBRATION

Every module is individually factory calibrated at two temperatures and two pressures. As a result, 6 coefficients necessary to compensate for process variations and temperature variations are calculated and stored in the 128-bit PROM of each module. These bits (partitioned into 6 coefficients) must be read by the microcontroller software and used in the program converting D1 and D2 into compensated pressure and temperature values.

SERIAL INTERFACE

The MS5607-02BA has built in two types of serial interfaces: SPI and I^2C. Pulling the Protocol Select pin PS to low selects the SPI protocol, pulling PS to high activates the I^2C bus protocol.

Pin PS	Mode	Pins used
High	I^2C	SDA
Low	SPI	SDI, SDO, CSB

SPI MODE

The external microcontroller clocks in the data through the input SCLK (Serial CLocK) and SDI (Serial Data In). In the SPI mode module can accept both mode 0 and mode 3 for the clock polarity and phase. The sensor responds on the output SDO (Serial Data Out). The pin CSB (Chip Select) is used to enable/disable the interface, so that other devices can talk on the same SPI bus. The CSB pin can be pulled high after the command is sent or after the end of the command execution (for example end of conversion). The best noise performance from the module is obtained when the SPI bus is idle and without communication to other devices during the ADC conversion.

I²C MODE

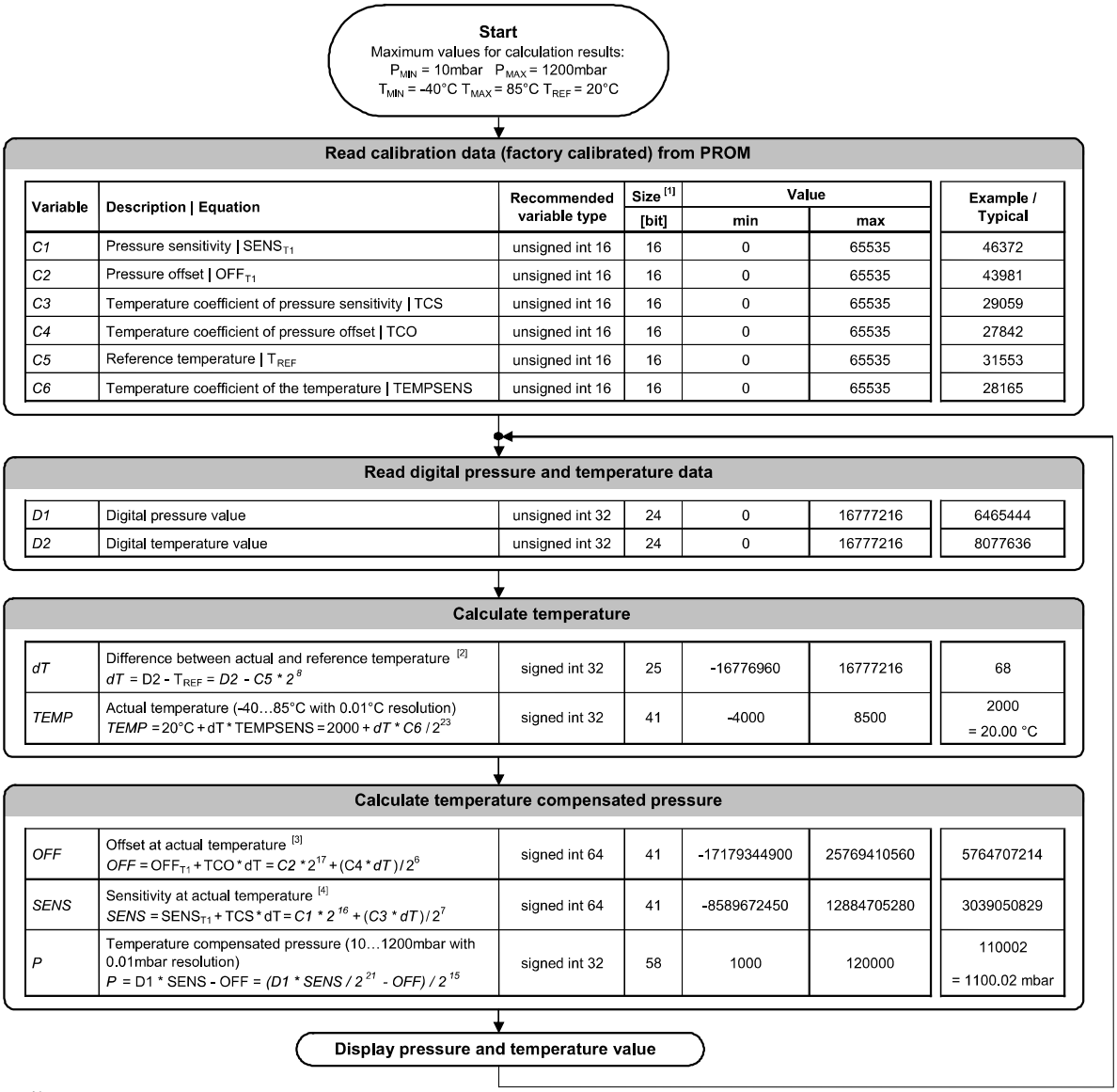
The external microcontroller clocks in the data through the input SCLK (Serial CLocK) and SDA (Serial DAta). The sensor responds on the same pin SDA which is bidirectional for the I²C bus interface. So this interface type uses only 2 signal lines and does not require a chip select, which can be favourable to reduce board space. In I²C-Mode the complement of the pin CSB (Chip Select) represents the LSB of the I²C address. It is possible to use two sensors with two different addresses on the I²C bus. The pin CSB shall be connected to VDD or GND (do not leave unconnected!).

COMMANDS

The MS5607-02BA has only five basic commands:

1. Reset
2. Read PROM (128 bit of calibration words)
3. D1 conversion
4. D2 conversion
5. Read ADC result (24 bit pressure / temperature)

PRESSURE AND TEMPERATURE CALCULATION



Variable	Description | Equation	Recommended variable type	Size [1] [bit]	Value min	Value max	Example / Typical
C1	Pressure sensitivity | $SENS_{T1}$	unsigned int 16	16	0	65535	46372
C2	Pressure offset | OFF_{T1}	unsigned int 16	16	0	65535	43981
C3	Temperature coefficient of pressure sensitivity | TCS	unsigned int 16	16	0	65535	29059
C4	Temperature coefficient of pressure offset | TCO	unsigned int 16	16	0	65535	27842
C5	Reference temperature | T_{REF}	unsigned int 16	16	0	65535	31553
C6	Temperature coefficient of the temperature | TEMPSENS	unsigned int 16	16	0	65535	28165
D1	Digital pressure value	unsigned int 32	24	0	16777216	6465444
D2	Digital temperature value	unsigned int 32	24	0	16777216	8077636
dT	Difference between actual and reference temperature [2] $dT = D2 - T_{REF} = D2 - C5 * 2^8$	signed int 32	25	-16776960	16777216	68
TEMP	Actual temperature (-40…85°C with 0.01°C resolution) $TEMP = 20°C + dT * TEMPSENS = 2000 + dT * C6 / 2^{23}$	signed int 32	41	-4000	8500	2000 = 20.00 °C
OFF	Offset at actual temperature [3] $OFF = OFF_{T1} + TCO * dT = C2 * 2^{17} + (C4 * dT) / 2^6$	signed int 64	41	-17179344900	25769410560	5764707214
SENS	Sensitivity at actual temperature [4] $SENS = SENS_{T1} + TCS * dT = C1 * 2^{16} + (C3 * dT) / 2^7$	signed int 64	41	-8589672450	12884705280	3039050829
P	Temperature compensated pressure (10…1200mbar with 0.01mbar resolution) $P = D1 * SENS - OFF = (D1 * SENS / 2^{21} - OFF) / 2^{15}$	signed int 32	58	1000	120000	110002 = 1100.02 mbar

Notes
[1] Maximal size of intermediate result during evaluation of variable
[2] min and max have to be defined
[3] min and max have to be defined
[4] min and max have to be defined

Figure 2: Flow chart for pressure and temperature reading and software compensation.

SPI INTERFACE

COMMANDS

Size of each command is 1 byte (8 bits) as described in the table below. After ADC read commands the device will return 24 bit result and after the PROM read 16bit result. The address of the PROM is embedded inside of the PROM read command using the a2, a1 and a0 bits.

	Command byte								hex value
Bit number	0	1	2	3	4	5	6	7	
Bit name	PRM	COV	-	Typ	Ad2/ Os2	Ad1/ Os1	Ad0/ Os0	Stop	
Command									
Reset	0	0	0	1	1	1	1	0	0x1E
Convert D1 (OSR=256)	0	1	0	0	0	0	0	0	0x40
Convert D1 (OSR=512)	0	1	0	0	0	0	1	0	0x42
Convert D1 (OSR=1024)	0	1	0	0	0	1	0	0	0x44
Convert D1 (OSR=2048)	0	1	0	0	0	1	1	0	0x46
Convert D1 (OSR=4096)	0	1	0	0	1	0	0	0	0x48
Convert D2 (OSR=256)	0	1	0	1	0	0	0	0	0x50
Convert D2 (OSR=512)	0	1	0	1	0	0	1	0	0x52
Convert D2 (OSR=1024)	0	1	0	1	0	1	0	0	0x54
Convert D2 (OSR=2048)	0	1	0	1	0	1	1	0	0x56
Convert D2 (OSR=4096)	0	1	0	1	1	0	0	0	0x58
ADC Read	0	0	0	0	0	0	0	0	0x00
PROM Read	1	0	1	0	Ad2	Ad1	Ad0	0	0xA0 to 0xAE

Figure 4: Command structure

RESET SEQUENCE

The Reset sequence shall be sent once after power-on to make sure that the calibration PROM gets loaded into the internal register. It can be also used to reset the device ROM from an unknown condition

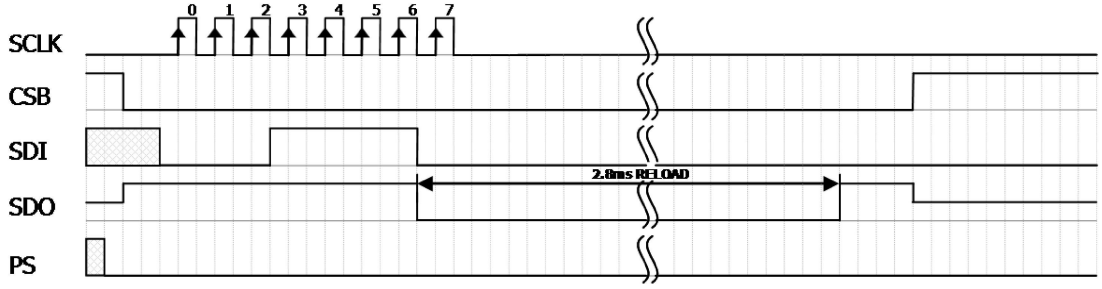


Figure 5: Reset command sequence SPI mode 0

APPLICATION CIRCUIT

The MS5607-02BA is a circuit that can be used in conjunction with a microcontroller in mobile altimeter applications. It is designed for low-voltage systems with a supply voltage of 3 V.

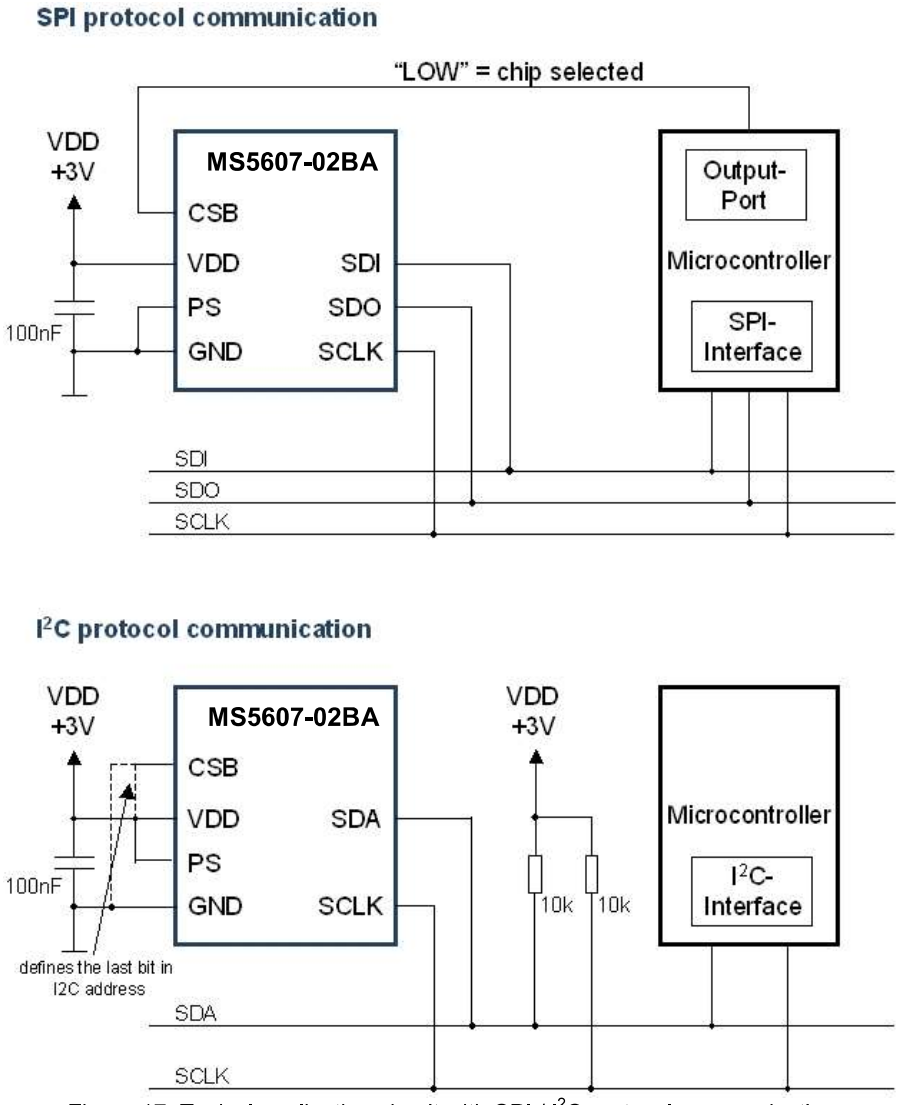


Figure 17: Typical application circuit with SPI / I^2C protocol communication

Anexo V. Datasheet Sensor de Temperatura y Humedad

Si7006-A20

I²C Humidity and Temperature Sensor

Features

- Precision Relative Humidity Sensor
  - ± 5% RH (max), 0–90% RH
- High Accuracy Temperature Sensor
  - ±1 °C (max), –10 to 85 °C
- 0 to 100% RH operating range
- –40 to +125 °C operating range
- Wide operating voltage (1.9 to 3.6 V)
- Low Power Consumption
  - 150 µA active current
  - 60 nA standby current
- Factory-calibrated
- I²C Interface
- Integrated on-chip heater
- 3x3 mm DFN Package
- Excellent long term stability
- Optional factory-installed cover
  - Low-profile
  - Protection during reflow
  - Excludes liquids and particulates

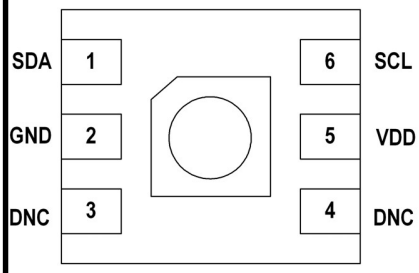
Ordering Information:
See page 28.

Applications

- HVAC/R
- Thermostats/humidistats
- Respiratory therapy
- White goods
- Indoor weather stations
- Micro-environments/data centers
- Automotive climate control and defogging
- Asset and goods tracking
- Mobile phones and tablets

Pin Assignments

Top View



Patent Protected. Patents pending

Description

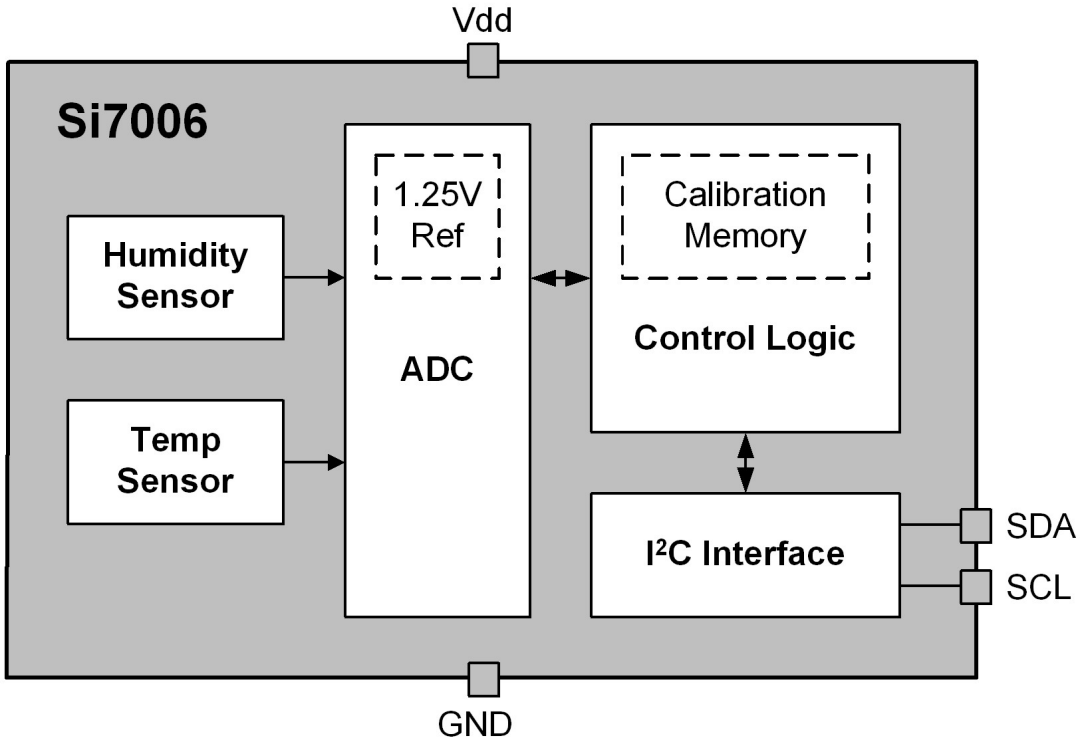
The Si7006 I²C Humidity and Temperature Sensor is a monolithic CMOS IC integrating humidity and temperature sensor elements, an analog-to-digital converter, signal processing, calibration data, and an I²C Interface. The patented use of industry-standard, low-K polymeric dielectrics for sensing humidity enables the construction of low-power, monolithic CMOS Sensor ICs with low drift and hysteresis, and excellent long term stability.

The humidity and temperature sensors are factory-calibrated and the calibration data is stored in the on-chip non-volatile memory. This ensures that the sensors are fully interchangeable, with no recalibration or software changes required.

The Si7006 is available in a 3x3 mm DFN package and is reflow solderable. It can be used as a hardware- and software-compatible drop-in upgrade for existing RH/temperature sensors in 3x3 mm DFN-6 packages, featuring precision sensing over a wider range and lower power consumption. The optional factory-installed cover offers a low profile, convenient means of protecting the sensor during assembly (e.g., reflow soldering) and throughout the life of the product, excluding liquids (hydrophobic/oleophobic) and particulates.

The Si7006 offers an accurate, low-power, factory-calibrated digital solution ideal for measuring humidity, dew-point, and temperature, in applications ranging from HVAC/R and asset tracking to industrial and consumer platforms.

Functional Block Diagram



1. Electrical Specifications

Unless otherwise specified, all min/max specifications apply over the recommended operating conditions.

Table 1. Recommended Operating Conditions

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
Power Supply	V_{DD}		1.9		3.6	V
Operating Temperature	T_A		–40	—	+125	°C

Table 2. General Specifications

1.9 ≤ V_{DD} ≤ 3.6 V; T_A = –40 to 125 °C default conversion time unless otherwise noted.

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
Input Voltage High	V_{IH}	SCL, SDA pins	0.7xV_{DD}	—	—	V
Input Voltage Low	V_{IL}	SCL, SDA pins	—	—	0.3xV_{DD}	V
Input Voltage Range	V_{IN}	SCL, SDA pins with respect to GND	0.0	—	V_{DD}	V
Input Leakage	I_{IL}	SCL, SDA pins	—	—	1	µA
Output Voltage Low	V_{OL}	SDA pin; I_{OL} = 2.5 mA; V_{DD} = 3.3 V	—	—	0.6	V
		SDA pin; I_{OL} = 1.2 mA; V_{DD} = 1.9 V	—	—	0.4	V
Current Consumption	I_{DD}	RH conversion in progress	—	150	180	µA
		Temperature conversion in progress	—	90	120	µA
		Standby, –40 to +85 °C[2]	—	0.06	0.62	µA
		Standby, –40 to +125 °C[2]	—	0.06	3.8	µA
		Peak IDD during powerup[3]	—	3.5	4.0	mA
		Peak IDD during I^2C operations[4]	—	3.5	4.0	mA
Heater Current[5]	I_{HEAT}		—	3.1 to 94.2	—	mA

Notes:

1. Initiating a RH measurement will also automatically initiate a temperature measurement. The total conversion time will be t_{CONV}(RH) + t_{CONV}(T).
2. No conversion or I^2C transaction in progress. Typical values measured at 25 °C.
3. Occurs once during powerup. Duration is <5 msec.
4. Occurs during I^2C commands for Reset, Read/Write User Registers, Read EID, and Read Firmware Version. Duration is <100 µs when I^2C clock speed is >100 kHz (>200 kHz for 2-byte commands).
5. Additional current consumption when HTRE bit enabled. See Section “5.5. Heater” for more information.

Table 2. General Specifications (Continued)

1.9 ≤ V_{DD} ≤ 3.6 V; T_A = –40 to 125 °C default conversion time unless otherwise noted.

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
Conversion Time[1]	t_{CONV}	12-bit RH	—	10	12	ms
		11-bit RH	—	5.8	7	
		10-bit RH	—	3.7	4.5	
		8-bit RH	—	2.6	3.1	
		14-bit temperature	—	7	10.8	
		13-bit temperature	—	4	6.2	
		12-bit temperature	—	2.4	3.8	
		11-bit temperature	—	1.5	2.4	
Powerup Time	t_{PU}	From V_{DD} ≥ 1.9 V to ready for a conversion, 25 °C	—	18	25	ms
		From V_{DD} ≥ 1.9 V to ready for a conversion, full temperature range	—	—	80	
		After issuing a software reset command	—	5	15	

Notes:
1. Initiating a RH measurement will also automatically initiate a temperature measurement. The total conversion time will be t_{CONV}(RH) + t_{CONV}(T).
2. No conversion or I^2C transaction in progress. Typical values measured at 25 °C.
3. Occurs once during powerup. Duration is <5 msec.
4. Occurs during I^2C commands for Reset, Read/Write User Registers, Read EID, and Read Firmware Version. Duration is <100 µs when I^2C clock speed is >100 kHz (>200 kHz for 2-byte commands).
5. Additional current consumption when HTRE bit enabled. See Section “5.5. Heater” for more information.

Table 3. I^2C Interface Specifications[1]

1.9 ≤ V_{DD} ≤ 3.6 V; T_A = –40 to +125 °C unless otherwise noted.

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
Hysteresis	V_{HYS}	High-to-low versus low-to-high transition	0.05 x V_{DD}	—	—	V
SCLK Frequency[2]	f_{SCL}		—	—	400	kHz
SCL High Time	t_{SKH}		0.6	—	—	µs
SCL Low Time	t_{SKL}		1.3	—	—	µs
Start Hold Time	t_{STH}		0.6	—	—	µs
Start Setup Time	t_{STS}		0.6	—	—	µs

Notes:
1. All values are referenced to V_{IL} and/or V_{IH}.
2. Depending on the conversion command, the Si7006 may hold the master during the conversion (clock stretch). At above 100 kHz SCL, the Si7006 may also hold the master briefly for user register and device ID transactions. At the highest I^2C speed of 400 kHz the stretching will be <50 µs.
3. Pulses up to and including 50 ns will be suppressed.

Table 3. I²C Interface Specifications[1] (Continued)

1.9 ≤ V_{DD} ≤ 3.6 V; T_A = –40 to +125 °C unless otherwise noted.

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
Stop Setup Time	t_{SPS}		0.6	—	—	µs
Bus Free Time	t_{BUF}	Between Stop and Start	1.3	—	—	µs
SDA Setup Time	t_{DS}		100	—	—	ns
SDA Hold Time	t_{DH}		100	—	—	ns
SDA Valid Time	$t_{VD;DAT}$	From SCL low to data valid	—	—	0.9	µs
SDA Acknowledge Valid Time	$t_{VD;ACK}$	From SCL low to data valid	—	—	0.9	µs
Suppressed Pulse Width[3]	t_{SPS}		50	—	—	ns

Notes:
1. All values are referenced to V_{IL} and/or V_{IH}.
2. Depending on the conversion command, the Si7006 may hold the master during the conversion (clock stretch). At above 100 kHz SCL, the Si7006 may also hold the master briefly for user register and device ID transactions. At the highest I²C speed of 400 kHz the stretching will be <50 µs.
3. Pulses up to and including 50 ns will be suppressed.

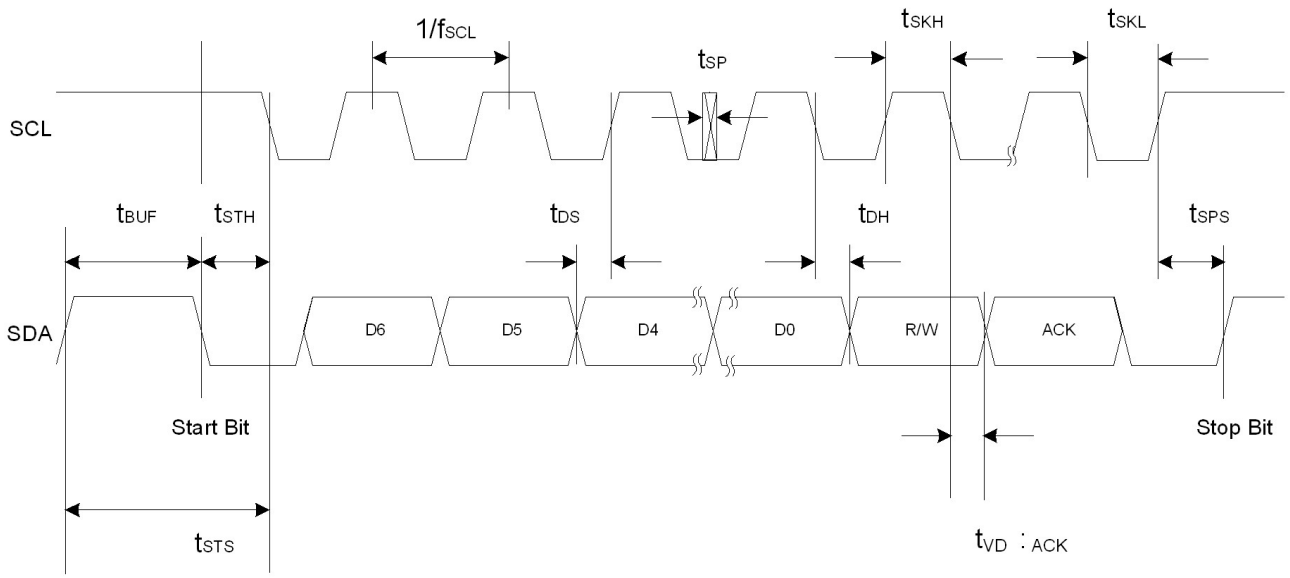


Figure 1. I²C Interface Timing Diagram

Table 4. Humidity Sensor

1.9 ≤ V_{DD} ≤ 3.6 V; T_A = 30 °C; default conversion time unless otherwise noted.

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
Operating Range[1]		Non-condensing	0	—	100	%RH
Accuracy[2, 3]		0 – 90% RH	—	±4	±5	%RH
		90 – 100% RH	See Figure 2.			
Repeatability/Noise		12-bit resolution	—	0.025	—	%RH RMS
		11-bit resolution	—	0.05	—	
		10-bit resolution	—	0.1	—	
		8-bit resolution	—	0.2	—	
Response Time[4]	$\tau_{63\%}$	1 m/s airflow, with cover	—	18	—	S
		1 m/s airflow, without cover	—	17	—	
Drift vs. Temperature			—	0.05	—	%RH/°C
Hysteresis			—	±1	—	%RH
Long Term Stability[3]			—	≤ 0.25	—	%RH/yr

Notes:

1. Recommended humidity operating range is 20% to 80% RH (non-condensing) over –10 °C to 60 °C. Prolonged operation beyond these ranges may result in a shift of sensor reading, with slow recovery time.
2. Excludes hysteresis, long term drift, and certain other factors and is applicable to non-condensing environments only. See section “4.1. Relative Humidity Sensor Accuracy” for more details.
3. Drift due to aging effects at typical room conditions of 30 °C and 30% to 50% RH. May be impacted by dust, vaporized solvents or other contaminants, e.g., out-gassing tapes, adhesives, packaging materials, etc. See section “4.7. Long Term Drift/Aging” .
4. Response time to a step change in RH. Time for the RH output to change by 63% of the total RH change.

2. Typical Application Circuits

The primary function of the Si7006 is to measure relative humidity and temperature. Figure 4 demonstrates the typical application circuit to achieve these functions.

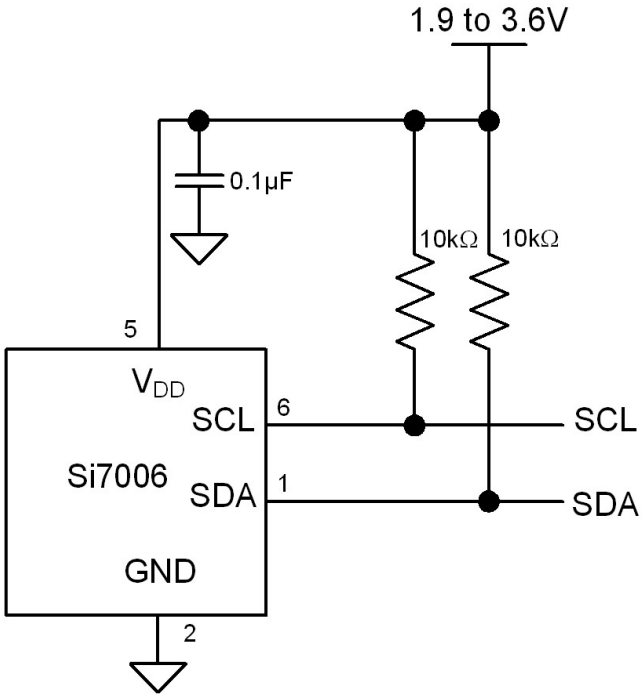


Figure 4. Typical Application Circuit for Relative Humidity and Temperature Measurement

5. I²C Interface

The Si7006 communicates with the host controller over a digital I²C interface. The 7-bit base slave address is 0x40. When sending commands to the device, the R/W bit is set high for a read command and low for a write command.

Table 10. I²C Slave Address Byte

A6	A5	A4	A3	A2	A1	A0	R/W
1	0	0	0	0	0	0	0

Master I²C devices communicate with the Si7006 using a command structure. The commands are listed in the I²C command table. Commands other than those documented below are undefined and should not be sent to the device.

Table 11. I²C Command Table

Command Description	Command Code
Measure Relative Humidity, Hold Master Mode	0xE5
Measure Relative Humidity, No Hold Master Mode	0xF5
Measure Temperature, Hold Master Mode	0xE3
Measure Temperature, No Hold Master Mode	0xF3
Read Temperature Value from Previous RH Measurement	0xE0
Reset	0xFE
Write RH/T User Register 1	0xE6
Read RH/T User Register 1	0xE7
Write Heater Control Register	0x51
Read Heater Control Register	0x11
Read Electronic ID 1st Byte	0xFA 0x0F
Read Electronic ID 2nd Byte	0xFC 0xC9
Read Firmware Revision	0x84 0xB8

5.1. Issuing a Measurement Command

The measurement commands instruct the Si7006 to perform one of two possible measurements; Relative Humidity or Temperature. The procedure to issue any one of these commands is identical. While the measurement is in progress, the option of either clock stretching (Hold Master Mode) or Not Acknowledging read requests (No Hold Master Mode) is available to indicate to the master that the measurement is in progress; the chosen command code determines which mode is used.

Optionally, a checksum byte can be returned from the slave for use in checking for transmission errors. The checksum byte will follow the least significant measurement byte if it is acknowledged by the master. The checksum byte is not returned if the master “not acknowledges” the least significant measurement byte. The checksum byte is calculated using a CRC generator polynomial of $x^8 + x^5 + x^4 + 1$, with an initialization of 0x00.

The checksum byte is optional after initiating an RH or temperature measurement with commands 0xE5, 0xF5, 0xE3, and 0xF3. The checksum byte is required for reading the electronic ID with commands 0xFA 0x0F and 0xFC 0xC9. For all other commands, the checksum byte is not supported.

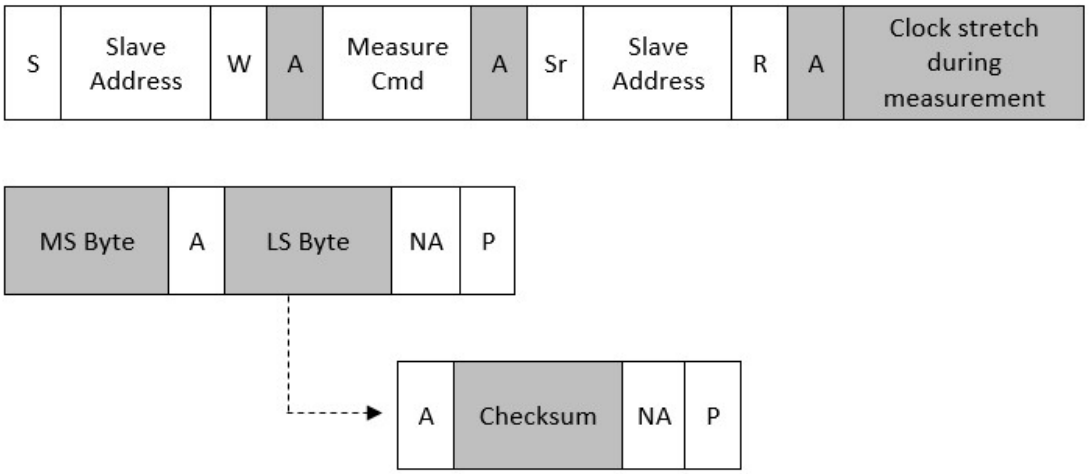
Table 12. I²C Bit Descriptions

Name	Symbol	Description
START	S	SDA goes low while SCL high
STOP	P	SDA goes high while SCL high
Repeated START	Sr	SDA goes low while SCL high. It is allowable to generate a STOP before the repeated start. SDA can transition to high before or after SCL goes high in preparation for generating the START.
READ	R	Read bit = 1
WRITE	W	Write bit = 0
All other bits	—	SDA value must remain high or low during the entire time SCL is high (this is the set up and hold time in Figure 1)

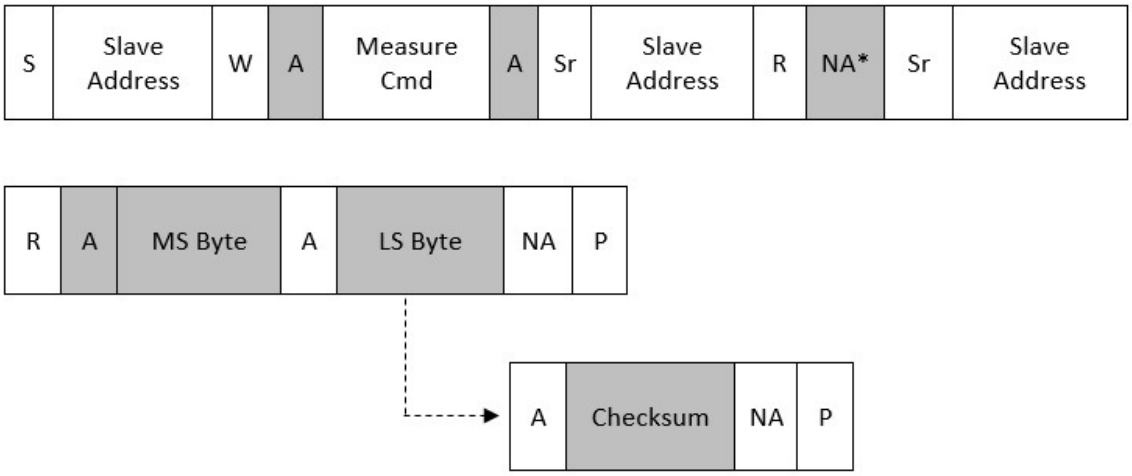
In the I²C sequence diagrams in the following sections, bits produced by the master and slave are color coded as shown:

Master	Slave

Sequence to perform a measurement and read back result (Hold Master Mode)



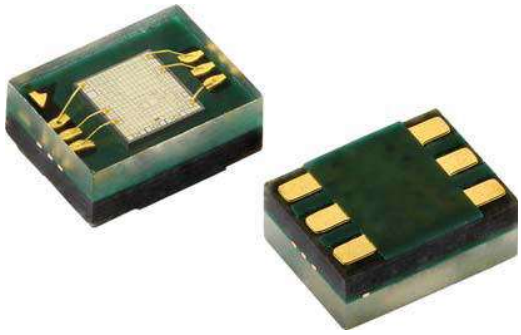
Sequence to perform a measurement and read back result (No Hold Master Mode)



***Note:** Device will NACK the slave address byte until conversion is complete.

Anexo VI. Datasheet Sensor UV

UV A Light Sensor with I²C Interface



FEATURES

- Package type: surface mount
- Dimensions (L x W x H in mm): 2.35 x 1.8 x 1.0
- Integrated modules: ultraviolet sensor (UV), and signal conditioning IC
- Converts solar UV light intensity to digital data
- Excellent UV sensitivity and linearity via Filtron™ technology
- Excellent performance of UV radiation measurement under long time solar UV exposure
- Excellent temperature compensation
- High dynamic detection resolution
- Standard I²C protocol interface
- Support acknowledge feature (ACK)
- Immunity on fluorescent light flicker software shutdown mode control
- Package: OPLGA
- Temperature compensation: -40 °C to +85 °C
- Floor life: 168 h, MSL 3, according to J-STD-020
- Output type: I²C bus
- Operation voltage: 2.7 V to 5.5 V
- Material categorization: for definitions of compliance please see www.vishay.com/doc?99912

Pb-free

e4

RoHS COMPLIANT

HALOGEN FREE

GREEN (5-2008)

DESCRIPTION

VEML6070 is an advanced ultraviolet (UV) light sensor with I²C protocol interface and designed by the CMOS process. It is easily operated via a simple I²C command. The active acknowledge (ACK) feature with threshold windows setting allows the UV sensor to send out a UVI alert message. Under a strong solar UVI condition, the smart ACK signal can be easily implemented by the software programming.

VEML6070 incorporates a photodiode, amplifiers, and analog / digital circuits into a single chip. VEML6070's adoption of Filtron™ UV technology provides the best spectral sensitivity to cover UV spectrum sensing. It has an excellent temperature compensation and a robust refresh rate setting that does not use an external RC low pass filter. VEML6070 has linear sensitivity to solar UV light and is easily adjusted by an external resistor. Software shutdown mode is provided, which reduces power consumption to be less than 1 µA. VEML6070's operating voltage ranges from 2.7 V to 5.5 V.

APPLICATIONS

- Solar UV indicator
- Cosmetic / outdoor sport handheld product
- Consumer products

PRODUCT SUMMARY

PART NUMBER	OPERATING VOLTAGE RANGE (V)	I²C BUS VOLTAGE RANGE (V)	PEAK SENSITIVITY (nm)	RANGE OF SPECTRAL BANDWIDTH $\lambda_{0.5}$ (nm)	OUTPUT CODE
VEML6070	2.7 to 5.5	1.7 to 5.5	355	± 20	16 bit, I²C

Note

(1) Adjustable through I²C interface

ORDERING INFORMATION

ORDERING CODE	PACKAGING	VOLUME (1)	REMARKS
VEML6070	Tape and reel	MOQ: 2500 pcs	2.35 mm x 1.8 mm x 1.0 mm

Note

(1) MOQ: minimum order quantity

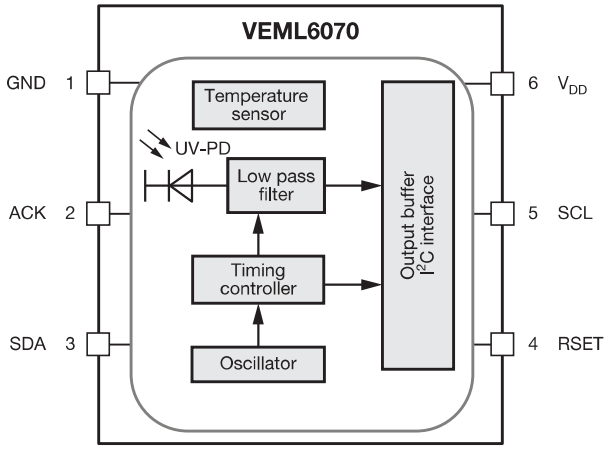
ABSOLUTE MAXIMUM RATINGS (T_{amb} = 25 °C, unless otherwise specified)

PARAMETER	TEST CONDITION	SYMBOL	MIN.	MAX.	UNIT
Supply voltage		V_{DD}	0	6.0	V
Operation temperature range		T_{amb}	-40	+85	°C

RECOMMENDED OPERATING CONDITIONS (T_{amb} = 25 °C, unless otherwise specified)					
PARAMETER	TEST CONDITION	SYMBOL	MIN.	MAX.	UNIT
Supply voltage		V_{DD}	2.7	5.5	V
Operation temperature range		T_{amb}	-40	+85	°C
I²C bus operating frequency		$f_{(I2CCLK)}$	10	400	kHz

PIN DESCRIPTIONS			
PIN ASSIGNMENT	SYMBOL	TYPE	FUNCTION
1	GND	I	Power supply ground, all voltage are reference to GND
2	ACK	O (open drain)	Acknowledge pin
3	SDA	I / O (open drain)	I²C digital serial data output to the host
4	SET		Light reading adjustment, connect a resistor to GND
5	SCL	I	I²C digital serial clock input from the host
6	V_{DD}	I	Supply voltage

BLOCK DIAGRAM

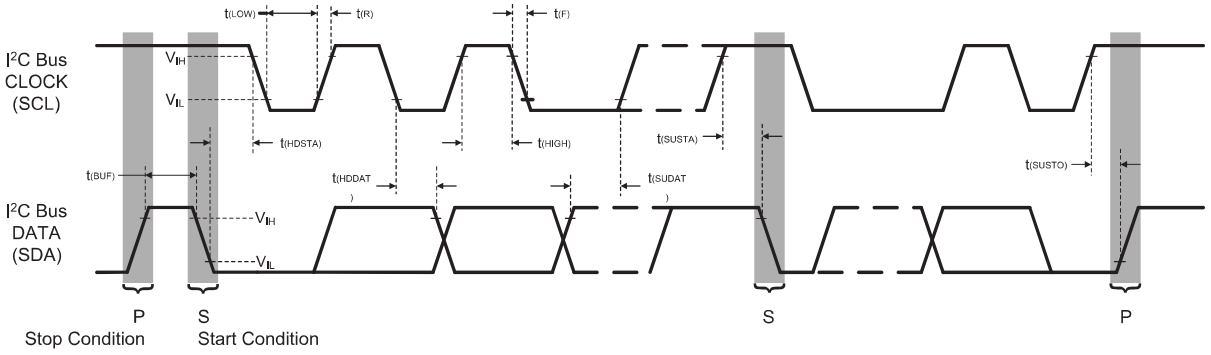


BASIC CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)							
PARAMETER		TEST CONDITION	SYMBOL	MIN.	TYP.	MAX.	UNIT
Supply operation voltage			V_{DD}	2.7	-	5.5	V
Supply current		R_{SET} = 240 kΩ (1)(2)	I_{DD}	-	100	250	µA
I²C signal input	Logic high	(1)(2)	V_{IH}	1.5	-	V_{DD}	V
	Logic low		V_{IL}	-	-	0.8	
Peak sensitivity wavelength			λ_p	-	355	-	nm
Range of spectral sensitivity			$\lambda_{0.1}$	320	-	410	nm
UVA sensitivity		R_{SET} = 240 kΩ, I_T = 1T (3)		-	5	-	µW/cm²/step
Maximum UVA detection power		R_{SET} = 240 kΩ, I_T = 1T (3)		-	-	328	mW/cm²
Dark offset		R_{SET} = 240 kΩ, I_T = 1T (1)		0	1	5	steps
Output offset		R_{SET} = 240 kΩ, I_T = 1T (1)(4)		-	2	-	steps
Shutdown current		Light condition = dark (1)	I_{DD}	-	1	15	µA

Notes

(1) Test condition: V_{DD} = 3.3 V, temperature: 25°C
(2) Light source: solar light source
(3) Test using 365 nm UVA LED
(4) Ambient light intensity = 500 lx

I²C TIMING CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)						
PARAMETER	**SYMBOL**	**STANDARD MODE**		**FAST MODE**		**UNIT**
		MIN.	**MAX.**	**MIN.**	**MAX.**	
Clock frequency	$f_{(SMBCLK)}$	10	100	10	400	kHz
Bus free time between start and stop condition	$t_{(BUF)}$	4.7	-	1.3	-	µs
Hold time after (repeated) start condition; after this period, the first clock is generated	$t_{(HDSTA)}$	4.0	-	0.6	-	µs
Repeated start condition setup time	$t_{(SUSTA)}$	4.7	-	0.6	-	µs
Stop condition setup time	$t_{(SUSTO)}$	4.0	-	0.6	-	µs
Data hold time	$t_{(HDDAT)}$		3450	-	900	ns
Data setup time	$t_{(SUDAT)}$	250	-	100	-	ns
I²C clock (SCK) low period	$t_{(LOW)}$	4.7	-	1.3	-	µs
I²C clock (SCK) high period	$t_{(HIGH)}$	4.0	-	0.6	-	µs
Detect clock / data low timeout	$t_{(TIMEOUT)}$	25	35	-	-	ms
Clock / data fall time	$t_{(F)}$	-	300	-	300	ns
Clock / data rise time	$t_{(R)}$	-	1000	-	300	ns



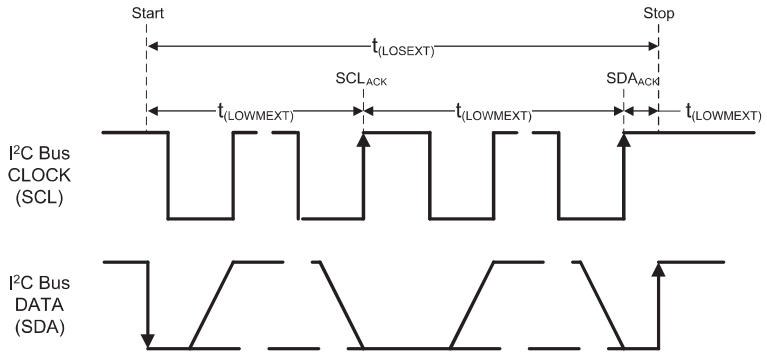


Fig. 1 - I²C Timing Diagram

Digital Interface

VEML6070 contains a 8-bit command register written via the I²C bus. All operations can be controlled by the command register. The simple command structure enables users to easily program the operation setting and latch the light data from VEML6070. In figure 9, VEML6070 I²C command format description for reading and writing operation between the host and VEML6070 are shown. The white sections indicate host activity and the gray sections indicate VEML6070's acknowledgement of the host access activity.

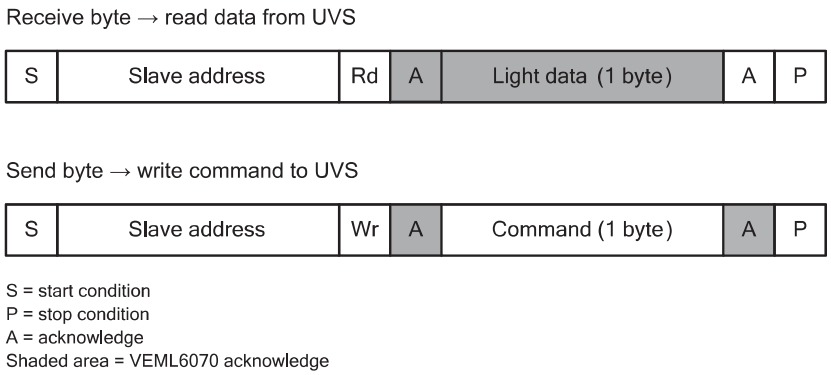


Fig. 9 - VEML6070 Command Protocol

Slave Address and Function Description

The VEML6070 has one slave address used for write functions (command) and two slave addresses used for read functions (UV data LSB and MSB).

The 7-bit address for write functions is 38h = 0111000**x** resulting in a 70h = 01110000 8-bit address. The 7-bit addresses for read functions are 38h = 0111000**x** for the UV Data LSB and 39h = 0111001**x** for the UV data MSB. This results in a 71h = 01110001 and 73h = 01110011 8-bit address, respectively. The 7-bit address 39h should not be used for a write function.

Command Register Format

VEML6070 provides a command to set device operations and sensitivity adjustment. This command is 8-bit long and includes 4 parameter groups for programming. The command format descriptions and register setting explanations are shown in tables 1 and 2.

TABLE 1 - COMMAND REGISTER BITS DESCRIPTION

COMMAND FORMAT							
Reserved		ACK	ACK_THD	IT		Reserved	SD
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	ACK	THD	IT1	IT0	1	SD

DESCRIPTION	
Reserved	Reserved
ACK	Acknowledge activity setting
ACK_THD	Acknowledge threshold window setting for byte mode usage
IT	Integration time setting
SD	Shutdown mode setting

TABLE 2 - REGISTER TABLE SETTING				
BITS SETTING	DESCRIPTION		BITS SETTING	DESCRIPTION
Reserved	Set initial value to (0 : 0)		(IT1 : IT0) (1)	(0 : 0) = ½T (0 : 1) = 1T (1 : 0) = 2T (1 : 1) = 4T
ACK	0 = disable		Reserved	Set initial value to 1
	1 = enable			
ACK_THD	0 = 102 steps		SD	0 = disable
	1 = 145 steps			1 = enable

Note

(1) Please refer to table 4, "Example of Refresh Time and R_{SET} Value Relation"

Data Access

VEML6070 has 16-bit resolution to give high resolution for light intensity sensing. Examples of the application setting are shown in table 3.

TABLE 3 - DATA ACCESS DESCRIPTION																
	VEML6070 16-BIT DATA BUFFER															
Data bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sequence 1	◄							►								
Sequence 2									◄							►

Notes

- Slave addresses (8 bits) for data read: 0x71 and 0x73
- Data reading sequence for the host:
  -Set read command to 0x73, read MSB 8 bits of 16 bits light data (sequence 1)
  -Set read command to 0x71, read LSB 8 bits of 16 bits light data for completing data structure (sequence 2)

Initialization

VEML6070 needs to be initialized while the system's power is on. The initialization includes two major steps: (1) clear ACK state of UVS and (2) fill the initial value, 06 (HEX), into the 0x70 addresses. After the initialization is completed, VEML6070 can be programmable for operation by write command setting from the host. VEML6070 initialization is recommended to be completed within 150 ms.

Acknowledge Activity

VEML6070 provides a function for sending an acknowledge signal (ACK) to the host when the value of sensed UV light is over the programmed threshold (ACK_THD) value. The purpose of the ACK signal is similar to the interrupt feature which informs the host once the sensed data level goes beyond the interrupt threshold setting. VEML6070 has two ACK threshold values, 102 steps and 145 steps.

There are two methods of driving acknowledge condition and read / write command to VEML6070:

(1) If the host implements the INT function, it performs a modified received byte operation to disengage VEML6070's acknowledge signal and acknowledge alert response address (ARA), 0x18 (Hex). A command format for responses to an ARA is shown in figure 10.

S	ARA (0x18)	Rd	A	UVS Slave Address	A	P

Fig. 10 - Command Format for Responds to an ARA

(2) If the host does not implement this feature, it should periodically access the ARA or read ARA before setting each read / write command.

The behavior of an ACK signal is similar to the INT definition in I^2C specification. For the hardware circuit design, this pin connects to an INT pin or GPIO pin of the MCU. The threshold ACK_THD definition is based on the sensitivity setting of VEML6070.

The ACK or UVI interrupt function allows the UVI sensing system to perform data pooling based on the interrupt event. The system sensor manager does not need to do continual data pooling and this significantly reduced the MCU loading. The ACK signal can also be used as a trigger event for popping up a warning UVI message.

Refresh Time Determination

VEML6070's refresh time can be determined by the R_{SET} value. Cooperating with the command register setting, the designer has a flexible way of defining the timing for light data collection. The default refresh time is 1T, (IT1 : IT0) = (0 : 1). If the R_{SET} value is changed, the default timing changes and the other parts in the register table also change by comparing itself with the default timing (refer to figure 7).

Table 4 is an example of two R_{SET} resistors that show the timing table that the system designer can use a flexible way to determine the desired refresh time.

TABLE 4 - EXAMPLE OF REFRESH TIME AND R_{SET} VALUE RELATION

REGISTER	SETTING	REFRESH TIME	
		R_{SET} = 300 kΩ	R_{SET} = 600 kΩ
(IT1 : IT0)	(0 : 0) = ½T	62.5 ms	125 ms
	(0 : 1) = 1T	125 ms	250 ms
	(1 : 0) = 2T	250 ms	500 ms
	(1 : 1) = 4T	500 ms	1000 ms

The designer can decide the refresh timing range requirement first, then choose an appropriate R_{SET} value for the timing range, and then write the correct value for the system application via I^2C protocol.

Designing the VEML6070 UV Light Sensor Into Applications

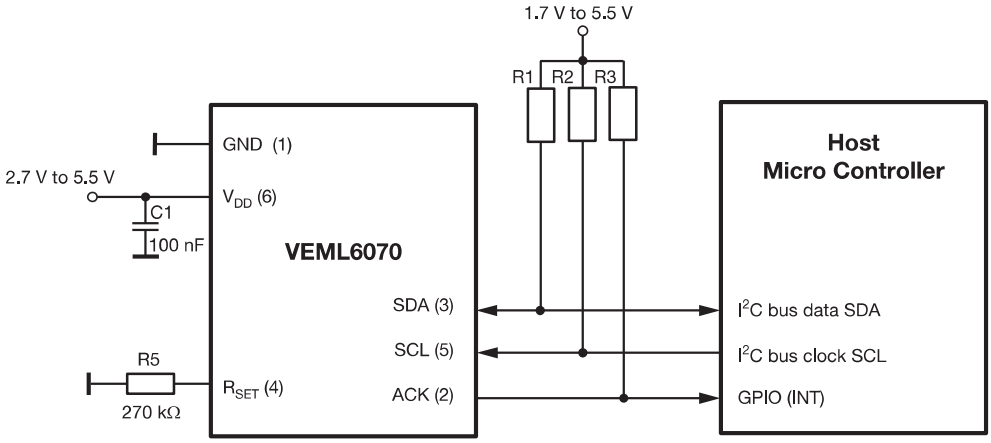


Fig. 2 - Application Circuit

The value for the pull-up resistors should be 2.2 kΩ.

The supply current of this device is also dependent on the R_{SET} value. When activated for measuring it is typically 100 µA; in shut-down mode (SD = 1) it is typically just 1 µA.

The resistor R_{SET} value at pin 4 of the VEML6070 needs to be selected depending on the application and required sensitivity. The table below shows how this value also affects the integration time that is programmed within the command register with bits 2 and 3 (IT0 and IT1).

EXAMPLE OF RELATION BETWEEN INTEGRATION TIME AND R_{SET} VALUE				
REGISTER	SETTING	REFRESH TIME		
		R_{SET} = 300 kΩ	R_{SET} = 600 kΩ	R_{SET} = 1.2 MΩ
(IT1 : IT0)	(0 : 0) = 1/2T	62.5 ms	125 ms	250 ms
	(0 : 1) = 1T	125 ms	250 ms	500 ms
	(1 : 0) = 2T	250 ms	500 ms	1000 ms
	(1 : 1) = 4T	500 ms	1000 ms	2000 ms

The VEML6070 shows its peak sensitivity at 355 nm. Bandwidth ($\lambda_{0.5}$) is achieved for a range of about 335 nm to 375 nm.

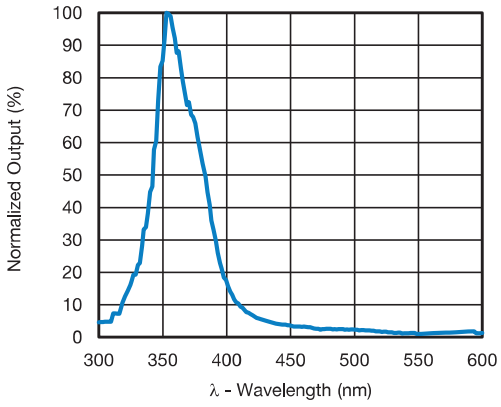


Fig. 3 - Relative Responsivity vs. Wavelength

APPLICATION NOTE

Designing the VEML6070 UV Light Sensor Into Applications

What does this wavelength mean? To understand this, the diagram below shows that it is in the middle of the so-called UVA region.

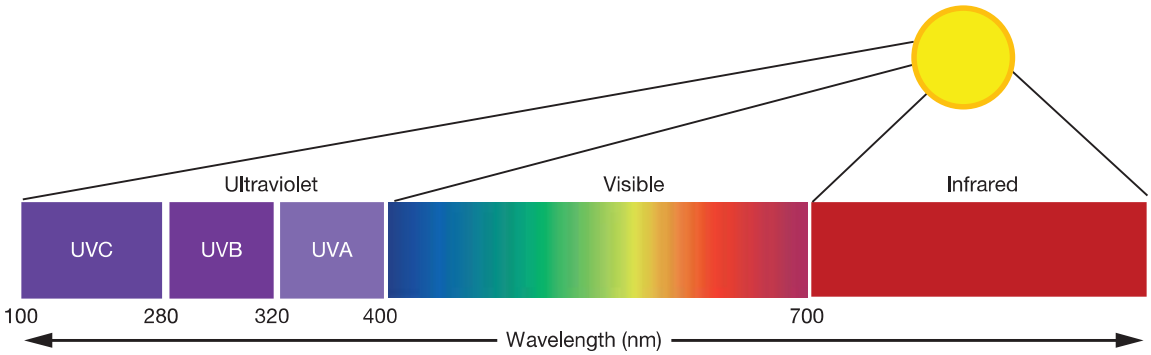


Fig. 4 - Light Spectrum

Visible light has wavelengths between 400 nm and 750 nm.
UV light has shorter wavelengths, from 200 nm to 400 nm.
UV type A has light with wavelengths between 320 nm and 400 nm.
UV type B has wavelengths between 280 and 320 nm.
UV type C is between 200 nm and 280 nm.

While UVA and UVB reach earth, UVC is blocked by our atmosphere, so it does not cause harm.

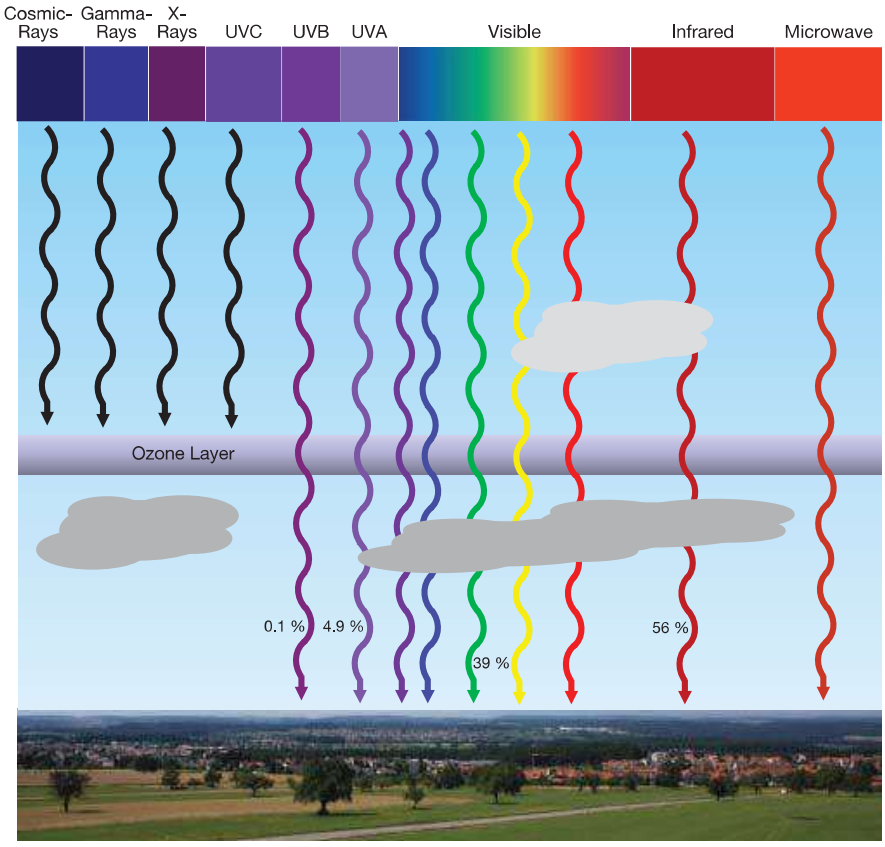


Fig. 5 - Radiation that Reaches the Earth's Surface

Designing the VEML6070 UV Light Sensor Into Applications

UVB rays - wavelengths ranging from 280 nm to 320 nm - are extremely energetic and harmful to the skin; they are responsible for 65 % of skin tumors. Thankfully, only 0.1 % of the solar energy that arrives on the earth's surface is in the form of UVB radiation.

UVA rays - wavelengths ranging from 320 nm to 400 nm - are less powerful than UVB rays, but are highly penetrating. They are capable of reaching the skin and are responsible for photoaging and the onset of different forms of skin cancer. 4.9 % of solar energy is made up of UVA rays.

In order to estimate the energy behind UV radiation and the risk level associated with it, the UV-index was established.

It is a quite complex calculation, weighted according to a curve and integrated over the whole spectrum. So, it cannot simply be related to the irradiance (measured in W/m^2).

The calculated index value appears on a scale of 0 to ≥ 11. This index scale is linear and its relation to irradiance strength is shown below.

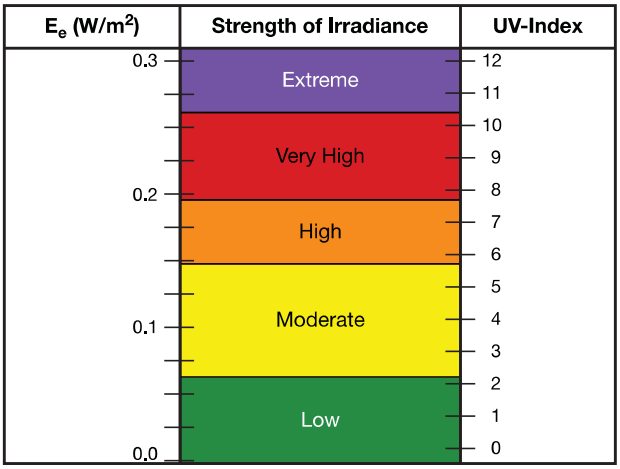


Fig. 6 - Strength of Irradiance and the UV-Index

In order to estimate the energy behind UV radiation and the risk level associated with it, the VEML6070 simply reads out the irradiance value and compares it with pre-defined values.

These given values are estimated, taking care to weigh the irradiance strength according to the wavelength and response performance of the VEML6070 (fig. 3).

Setting up and programming the VEML6070 is easily handled with just three I^2C-bus addresses: 0x70, 0x71, and 0x73.

TABLE 1 - VEML6070 SLAVE ADDRESS AND FUNCTION DESCRIPTION

SLAVE ADDRESS	OPERATION
0x70	Write command to VEML6070
0x72	Reserved
0x71	Read LSB 8 bits of VEML6070 ultraviolet light data
0x73	Read MSB 8 bits of VEML6070 ultraviolet light data

Designing the VEML6070 UV Light Sensor Into Applications

0x70 is the only command register where shut-down, integration time, and acknowledge activity settings are handled.

TABLE 2 - COMMAND REGISTER BITS DESCRIPTION							
COMMAND FORMAT							
Reserved		ACK	ACK_THD	IT		Reserved	SD
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	ACK	THD	IT1	IT0	1	SD

DESCRIPTION	
Reserved	Reserved
ACK	Acknowledge activity setting
ACK_THD	Acknowledge threshold window setting for byte mode usage
IT	Integration time setting
SD	Shutdown mode setting

As previously discussed, integration time also depends on the external resistor at pin 4. Together with a R_{SET} value of 270 kΩ, the table below shows UV light data values that lead to the UVI values shown on the left side.

UVI	R_{SET} = 270 kΩ; IT = 1T	R_{SET} = 270 kΩ; IT = 2T	R_{SET} = 270 kΩ; IT = 4T	UV-INDEX
≥ 11	≥ 2055	≥ 4109	≥ 8217	Extreme
8 to 10	1494 to 2054	2989 to 4108	5977 to 8216	Very High
6, 7	1121 to 1494	2242 to 2988	4483 to 5976	High
3 to 5	561 to 1120	1121 to 2241	2241 to 4482	Moderate
0 to 2	0 to 560	0 to 1120	0 to 2240	Low

As previously mentioned, other resistor values lead to other integration times with different output data. At 540 kΩ, the 270 kΩ output data values are doubled, and the 540 kΩ values are doubled at 1 MΩ, which means that the sensitivity is also doubled.

UVI	R_{SET} = 540 kΩ; IT = 1T	R_{SET} = 540 kΩ; IT = 2T	R_{SET} = 540 kΩ; IT = 4T	UV-INDEX
0 to 2	0 to 1120	0 to 2240	0 to 4480	Low
3 to 5	1121 to 2241	2241 to 4482	4481 to 8964	Moderate
6, 7	2242 to 2988	4483 to 5976	8965 to 11 952	High
8 to 10	2989 to 4108	5977 to 8216	11 953 to 16 432	Very High
≥ 11	≥ 4109	≥ 8217	≥ 16 433	Extreme

UVI	R_{SET} = 1 MΩ; IT = 1T	R_{SET} = 1 MΩ; IT = 2T	R_{SET} = 1 MΩ; IT = 4T	UV-INDEX
0 to 2	0 to 2241	0 to 4482	0 to 8964	Low
3 to 5	2242 to 4482	4483 to 8964	8965 to 17 928	Moderate
6, 7	4483 to 5976	8965 to 11 952	17 929 to 23 904	High
8 to 10	5977 to 8217	11 953 to 16434	23 905 to 32 868	Very High
≥ 11	≥ 8218	≥ 16 435	≥ 32 869	Extreme

VEML6070 light data is available at 0x71 (LSB) and 0x73 (MSB). Together they show the whole 16-bit value and report the present UV light conditions. This 16-bit value is transferred into decimal form and becomes the base for calculating the corresponding UVI.

As already stated, the above values are evaluated taking care to weigh the wavelength and response performance of the VEML6070. Factoring in the R_{SET} value and integration time leads to the UVI values.

Designing the VEML6070 UV Light Sensor Into Applications

WHAT VALUES MAY BE SEEN WITH THE VEML6070?

If no exact UV light source is available to check the performance of the VEML6070, and only “normal” daylight is being used to study the VEML6070’s response, please note that the UV power is dependent on the time of day, season, and location where the measurements will be performed. During the winter, the total skin-affecting irradiance may be so low that even within full sunshine no remarkable values will be seen.

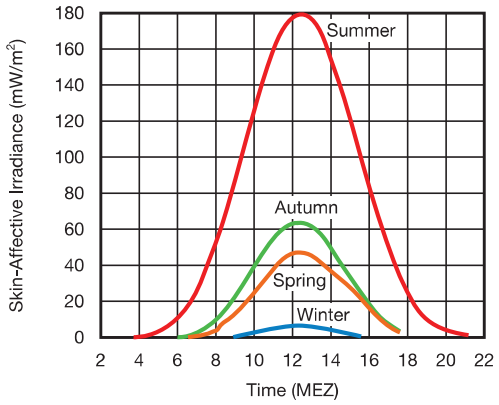


Fig. 7 - Skin Affecting Irradiance Level vs. Time Seen at the Beginning of the Four Seasons

MECHANICAL CONSIDERATIONS AND WINDOW CALCULATIONS FOR THE VEML6070

The UV sensor will be placed behind a window or cover. The window material should not only be completely transmissive to visible light (400 nm to 700 nm), but also at least to UVA wavelengths of 320 nm to 400 nm.

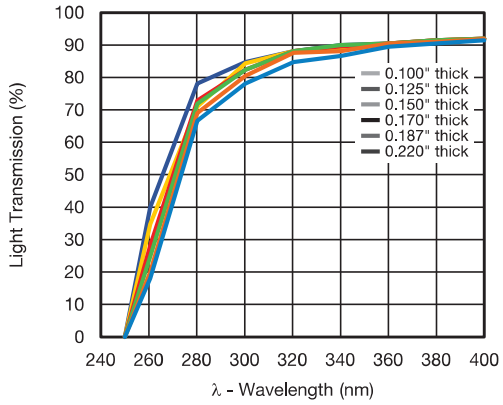
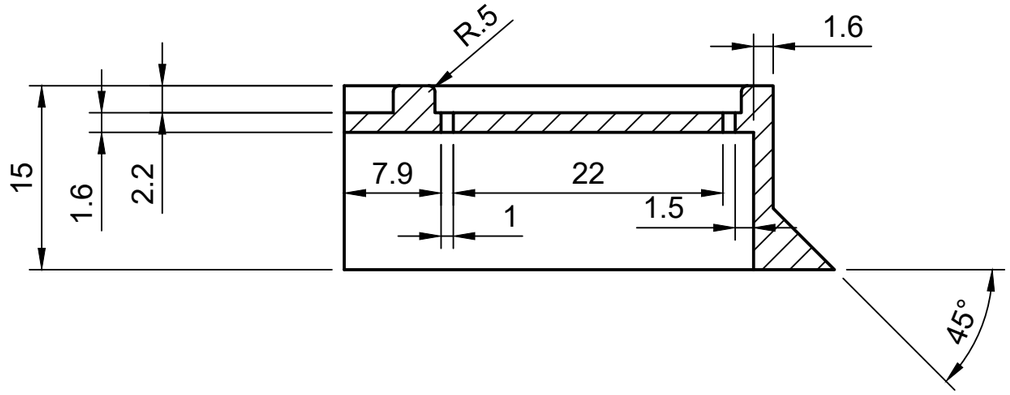


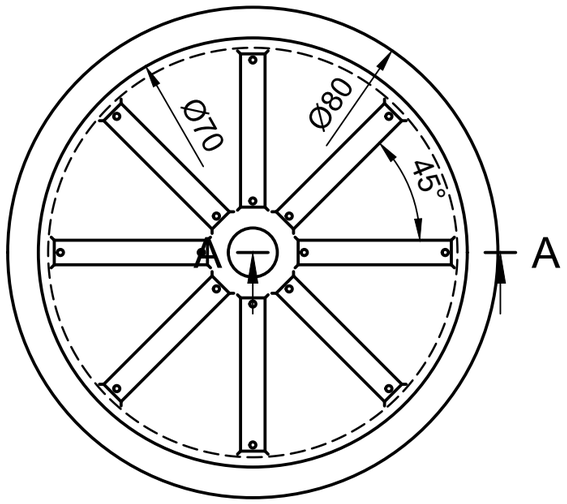
Fig. 8 - Light Transmission of ACRYLITE OP-4 Sheet
(www.aetnaplastics.com/site_media/media/documents/Acrylite_OP-4_Material_Data_Sheet.pdf)

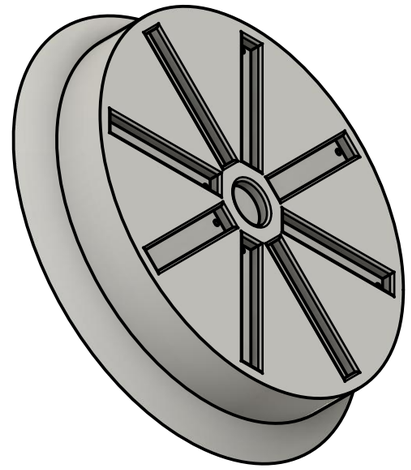
APPLICATION NOTE

Anexo VII. Planos Anemómetro y Veleta

A-A (2:1)

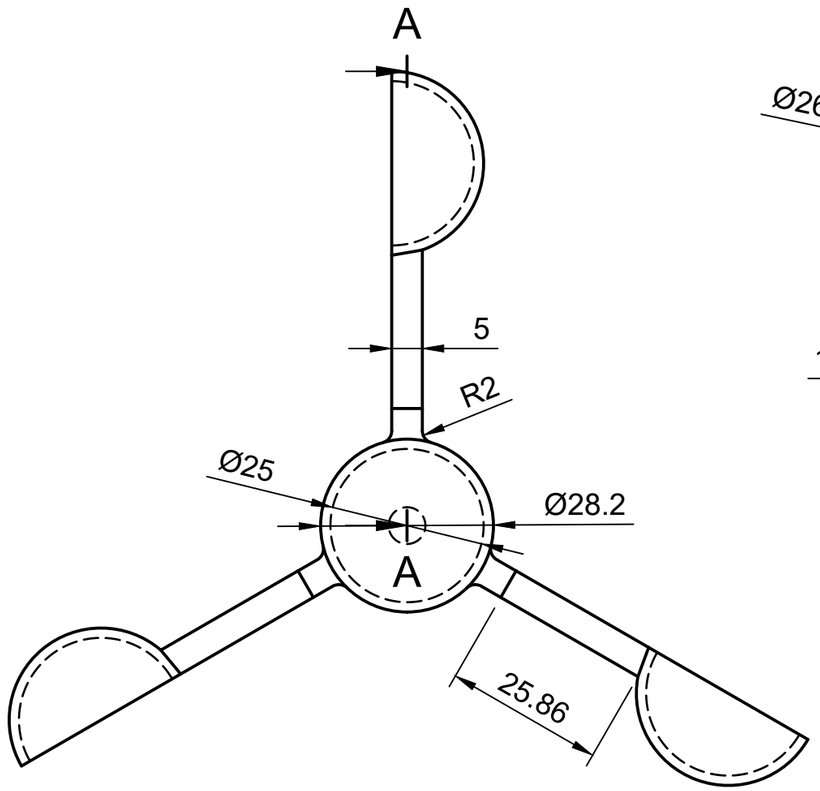


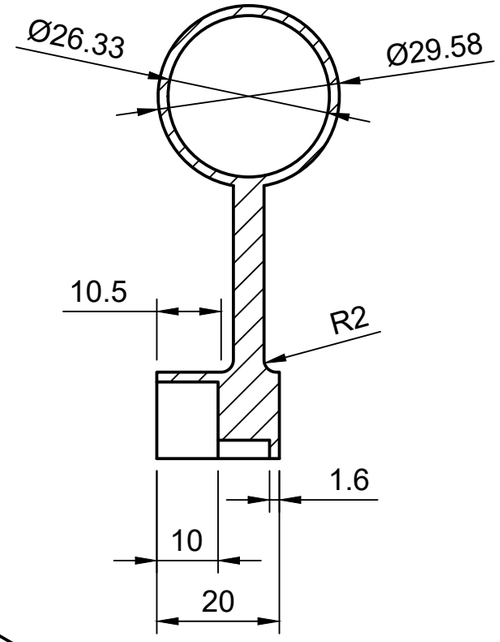


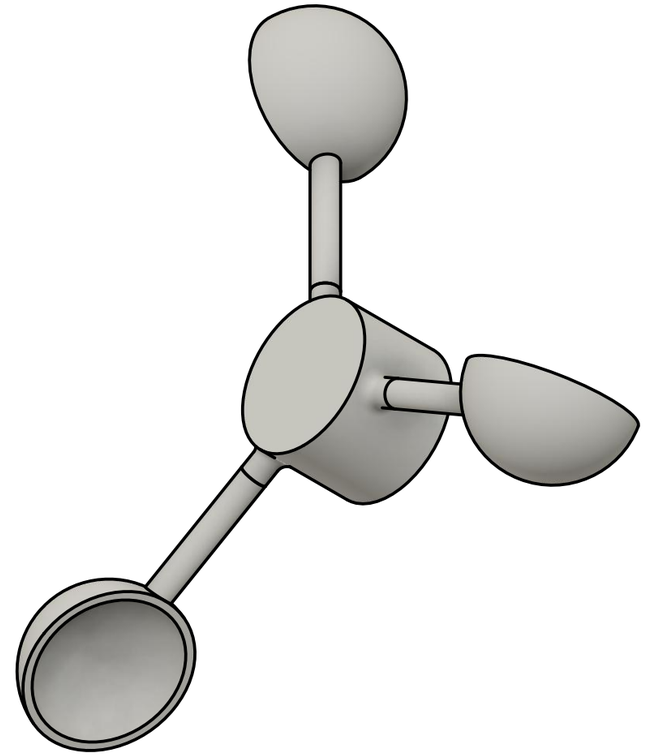


	Fecha:	Nombre:		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	**Pablo Pastor**		
Escala: 1:1	Título: Anemometro base			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: Anemometro 1

A-A (1:1)

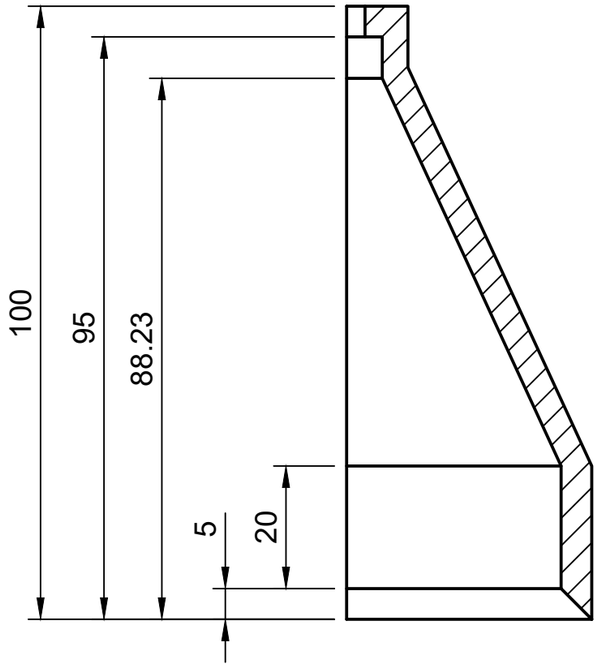


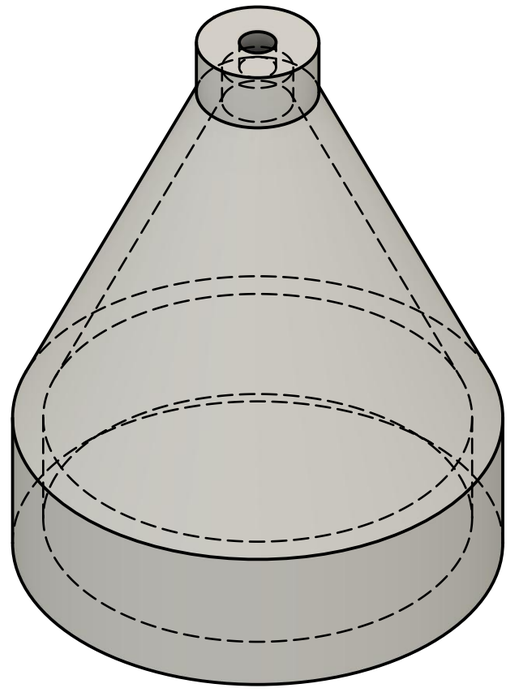


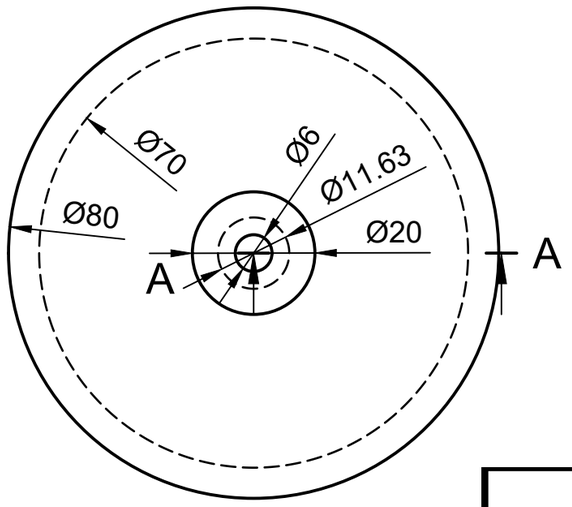


	Fecha:	Nombre:		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	**Pablo Pastor**		
Escala: 1:1	Título: Anemometro cazoletas			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: Anemometro 2

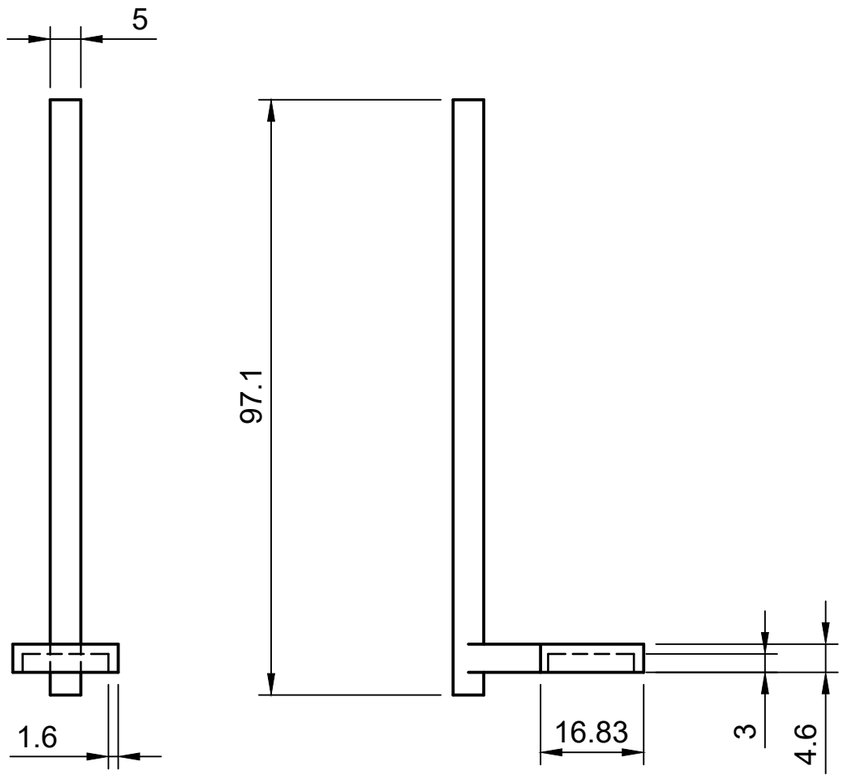
A-A (1:1)

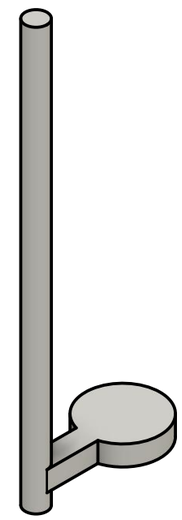


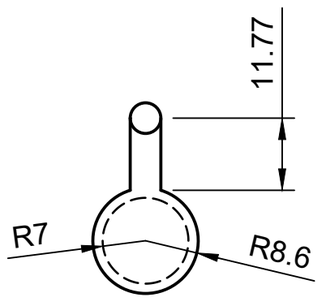




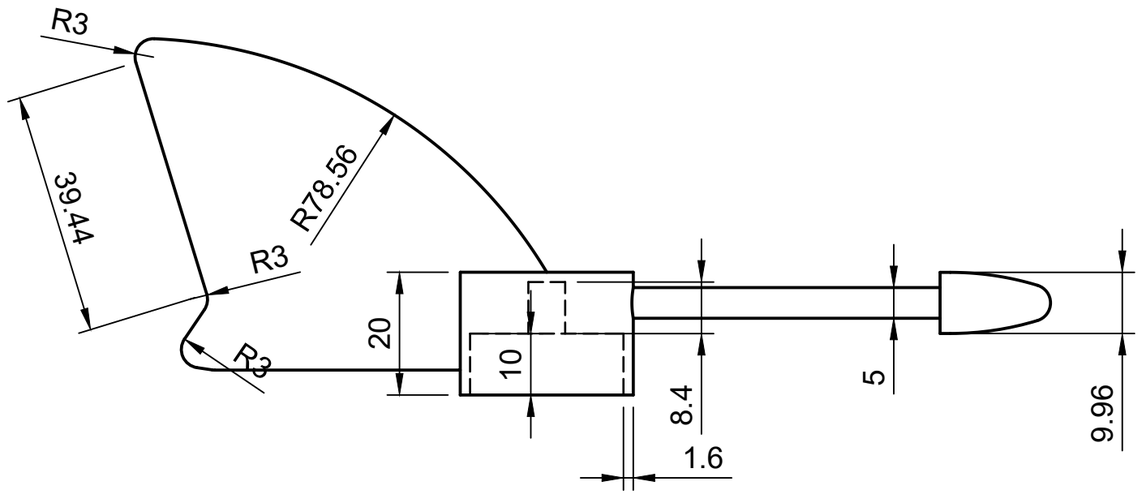
	Fecha:	Nombre:		
Dibujado:	08-18	**Pablo Pastor**		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Escala: 1:1	Título: Anemometro cono			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: Anemometro 3

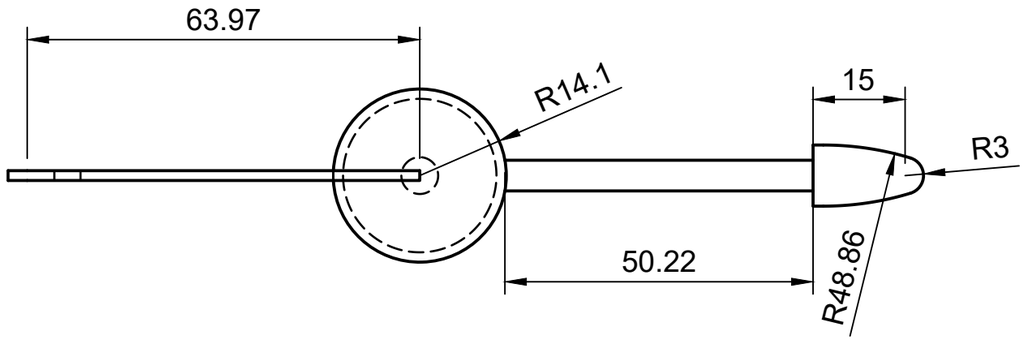


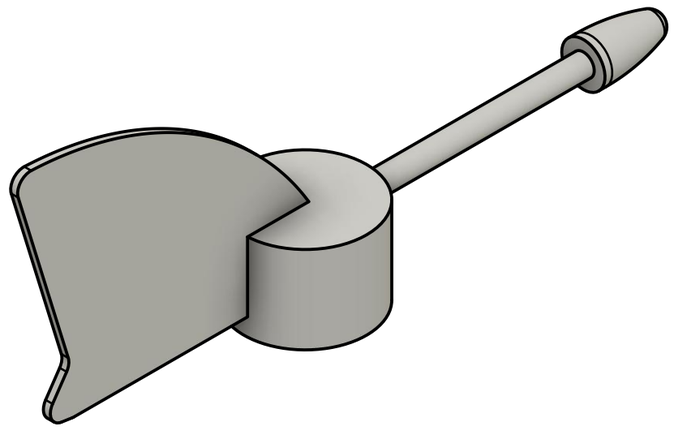




	Fecha:	Nombre:		
Dibujado:	08-18	**Pablo Pastor**		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Escala: 1:1	Título: Anemometro eje			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: Anemometro 4







	Fecha:	Nombre:		
Dibujado:	08-18	**Pablo Pastor**		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Escala: 1:1	Título: Anemometro veleta			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: Anemometro 5

Anexo VIII. Código implementado en el procesador QG8

El código se ha distribuido en el archivo principal y varias librerías. Los archivos descritos son:

- main.c
- i2c.h
- I2C.c
- ad.h
- AD.c
- sci.h
- SCI.c

main.c

```
/*****************************************************************
* Proyecto TFG
* Central meteorológica inalámbrica
*****************************************************************
* nombre fichero : main.c
*****************************************************************
* descripcion       : Codigo principal
* programador       : Pablo Pastor
* lenguaje          : ANSI C para CodeWarrior
* fecha             : 21 agosto 2018
*****************************************************************/

#include <hidef.h>                      // for EnableInterrupts macro
#include "derivative.h"                 // include peripheral declarations
#include "hcs08.h"                      // QG8 definitions
#include "ad.h"
#include "i2c.h"
#include "sci.h"

/*Variables globales*/
char Dir;
signed long P;
unsigned int RH, Temp, Batt, counter, Vel;
char UV, Irradiancia;


/*Funciones*/
void InitSystem(void);
void ConfigEntradas(void);

void main (void) {
  InitSystem();
  Init_AD();
  Init_SCI();
  Init_I2C();
  ConfigEntradas();
  EnableInterrupts;
```

```
  for(;;) {
      counter+=1;
      if (counter == 18000){ //cada 5 minutos
            counter=0;
            P = leer_presion();
            RH = leer_RH();
            Temp = leer_Temp();
            UV = leer_UV();
            Irradiancia = UV*25;
            Dir = leer_Dir();
            Vel = leer_Vel();
            Batt = leer_Battery();
            send_data(P,RH,Temp,UV,Irradiancia,Dir,Vel,Batt);
      }
      STOP; //Sleep mode
  }
}

void InitSystem(void) {
SOPT1_COPE = 0;                     // COP watchdog timer disabled
SOPT1_STOPE = 1;                    // Stop mode enabled
SOPT1_BKGDPE = 1;                   // Background DEbug Mode enabled -PTA4 reserved-
SOPT1_RSTPE = 0;                    // Pin reset disabled. PTA5 free.
SPMSC1 = 0;                         // disable LVD
SPMSC2 = 0;                         // stop3 mode selected
SRTISC = bRTIE | RTI_1024ms;  // enable RTI (internal clock source,
1024ms timeout, interrupt enabled)
}

void ConfigEntradas(void) {
      PTADD_PTADD0=0;         // PTA0 as input --- LDR/PRESSURE ---
      PTADD_PTADD1=0;         // PTA1 as input --- BATERIA ---
      PTADD_PTADD2=1;         // PTA2 as output--- SDA I2C H&T UV ---
      PTADD_PTADD3=1;         // PTA3 as output--- SCL I2C H&T UV---

      PTBDD_PTBDD0=1;         // PTB0 as output--- RX SERIAL ESP8266 ---
      PTBDD_PTBDD1=1;         // PTB1 as output--- TX SERIAL ESP8266 ---
      PTBDD_PTBDD2=0;         // PTB2 as input --- VEL/DIR WIND ---
      PTBDD_PTBDD3=1;         // PTB3 as output--- SW VEL ---
      PTBDD_PTBDD4=1;         // PTB4 as output--- SW ESP8266 ---
      PTBDD_PTBDD5=1;         // PTB5 as output--- SW DIR ---
      PTBDD_PTBDD6=1;         // PTB6 as output--- SW H&T ---
      PTBDD_PTBDD7=1;         // PTB7 as output--- SW LDR UV BATT PRESSURE---

 //SW Sensors Power Off
      PTBD_PTBD3=1;         //      --- SW VEL OFF ---
      PTBD_PTBD4=1;         //      --- SW ESP8266 OFF ---
      PTBD_PTBD5=1;         //      --- SW DIR OFF ---
      PTBD_PTBD6=1;         //      --- SW H&T OFF ---
      PTBD_PTBD7=1;         //      --- SW LDR UV BATT PRESSURE OFF---
}

void interrupt VectorNumber_Vrti rti_isr(void)
{                 // ISR for RTI interrupt (only clears RTIF flag)
  SRTISC_RTIACK = 1;    // acknowledge RTI interrupt (clear RTIF)
}
```

i2c.h

```
/*****************************************************************
* Proyecto TFG
* Central meteorológica inalámbrica
*****************************************************************
* nombre fichero : i2c.h
*****************************************************************
* descripción    : Fichero de cabecera modulo de comunicaciones i2c
* programador    : Pablo Pastor
* lenguaje       : ANSI C para CodeWarrior
* fecha          : 28 agosto 2017
*****************************************************************/

#ifndef i2c_h
#define i2c_h

void Init_I2C (void);
void byteWrite(char address, char data);
char byteRead(char address);
unsigned int byte2Read(char address);
long int byte3Read(char address);
signed long leer_presion(void);
unsigned int leer_RH(void);
unsigned int leer_Temp(void);
char leer_UV(void);
void retardo(unsigned int tiempo);

#endif
```

I2C.c

```
/*****************************************************************
* Proyecto TFG
* Central meteorológica inalámbrica
*****************************************************************
* nombre fichero : i2c.c
*****************************************************************
* descripción    : Implementación modulo comunicaciones i2c
* programador    : Pablo Pastor
* lenguaje       : ANSI C para CodeWarrior
* fecha          : 28 agosto 2017
*****************************************************************/

#include "derivative.h"
#include "i2c.h"
#include "hcs08.h"


void Init_I2C (void) {
      IICF_MULT = 2;     //Mul = 2;
      IICF_ICR = 0x0B;   //Divisor: 40
      // IIC Baud rate: bus speed /(mul*divider) = 8MHz/(2*40)= 100KHz
      // SDA Hold Time = bus period * SDA hold value = 1/8MHz * 9 =1.125uS
      IICC = bIICEN;     // enable the I2C interface
}
```

```
void byteWrite(char address, char data)
{
  do  //send the address field
  {
    // enter master transmit mode and send a start
    IICC = bIICEN | bTX | bMST;
    // send the address field
    IICD = address;
    while (!IICS_TCF);              // wait until the transmission ends

} while (IICS_RXAK);    // if no ACK was received, resend the address field

  do{
      IICD = data;
    while (!IICS_TCF);        // wait until the transmission ends
  }while(IICS_RXAK);

  IICC = bIICEN;    // send stop, exit master mode
}


char byteRead(char address)
{
  char data;
  while (IICS_BUSY);
  IICS_ARBL = 1;              // clear ARBL flag

  do  //send the address field
  {
    // enter master transmit mode and send a start
    IICC = bIICEN | bTX | bMST;
    IICD = address;
    while (!IICS_TCF);              // wait until the transmission ends

} while (IICS_RXAK);    // if no ACK was received, resend the address field

  IICC_TX = 0;                // receive mode
  IICC_TXAK = 1;              // send NACK in the next read
  data = IICD;
  while (!IICS_TCF);          // wait until the transmission ends
  IICC = bIICEN;              // send a stop, leave master mode
  return(data);
}

unsigned int byte2Read(char address)
{
  char data1, data2;
  unsigned int temp;
  while (IICS_BUSY);
  IICS_ARBL = 1;              // clear ARBL flag

  do  // send the address field
  {
    // enter master transmit mode and send a start
    IICC = bIICEN | bTX | bMST;
    // send the address field
    IICD = address;
    while (!IICS_TCF);              // wait until the transmission ends

} while (IICS_RXAK);    // if no ACK was received, resend the address field
```

```
  IICC_TX = 0;                // receive mode
  IICC_TXAK = 1;              // send NACK in the next read
  data1 = IICD;
  while (!IICS_TCF);          // wait until the transmission ends
  IICC_TXAK = 1;              // send NACK in the next read
  data2 = IICD;
  while (!IICS_TCF);          // wait until the transmission ends
  IICC = bIICEN;              // send a stop, leave master mode
  temp = (data1 << 8) | data2;
  return temp;
}

long int byte3Read(char address)
{
  char data1, data2, data3;
  long temp;
  while (IICS_BUSY);
  IICS_ARBL = 1;              // clear ARBL flag

  do  // first we send the address field
  {
    // enter master transmit mode and send a start
    IICC = bIICEN | bTX | bMST;
    // send the address field (R/W = write)
    IICD = address;
    while (!IICS_TCF);              // wait until the transmission ends

} while (IICS_RXAK);    // if no ACK was received, resend the address field

  IICC_TX = 0;                // receive mode
  IICC_TXAK = 1;              // send NACK in the next read
  data1 = IICD;
  while (!IICS_TCF);          // wait until the transmission ends
  IICC_TXAK = 1;              // send NACK in the next read
  data2 = IICD;
  while (!IICS_TCF);          // wait until the transmission ends
  IICC_TXAK = 1;              // send NACK in the next read
  data3 = IICD;
  while (!IICS_TCF);          // wait until the transmission ends
  IICC = bIICEN;              // send a stop, leave master mode
  temp = (data1 << 8) | data2;
  temp = (temp << 16) | data3;
  return temp;
}

signed long leer_presion(void)
{
    char presAdress = 0b11101111;
    unsigned int C1, C2, C3, C4, C5, C6;
    signed long D1, D2, dT, TEMP, press;
    signed long long OFF, SENS;
    PTBD_PTBD7=0;   //Sensor ON
    retardo(10);
    byteWrite(presAdress, 0xA0);
    C1 = byte2Read(presAdress);
    byteWrite(presAdress, 0xA2);
    C2 = byte2Read(presAdress);
    byteWrite(presAdress, 0xA4);
    C3 = byte2Read(0xA4);
    byteWrite(presAdress, 0xA6);
    C4 = byte2Read(presAdress);
```

```
    byteWrite(presAdress, 0xA8);
    C5 = byte2Read(presAdress);
    byteWrite(presAdress, 0xAA);
    C6 = byte2Read(presAdress);

    byteWrite(presAdress, 0x48); //Conversion pressure with OSR = 4096
    D1 = byte3Read(presAdress);

    byteWrite(presAdress, 0x58); //Conversion temperature with OSR = 4096
    D2 = byte3Read(presAdress);
    PTBD_PTBD7=1;   //Sensor OFF

    dT = D2 - (C5 << 8);  // = D2 - C5 * 2^8
    TEMP = 2000 + dT * (C6 >> 23); // =2000 + dT * C6 / 2^23
    OFF = (C2 << 17)  + ((C4 * dT) >> 6);
    SENS = (C1 << 16) + ((C3 * dT) >> 7);
    press = ((D1 * ( SENS >> 21) - OFF) >> 15);
    return(press);
}


unsigned int leer_RH(void)
{
      char RHAdressW = 0b10000000; // Write adress
      char RHAdressR = 0b10000001; // Read adress

      unsigned int RH;
      PTBD_PTBD6=0;       //Sensor ON
      retardo(10);
      byteWrite(RHAdressW, 0xF5); // Measure Comand
      RH = byte2Read(RHAdressR);
      PTBD_PTBD6=1;       //Sensor OFF
      RH = (unsigned int)(((long)(RH*125) >> 16) -6);
      return(RH);
}

unsigned int leer_Temp(void)
{
      char TempAdressW = 0b10000000; // Write adress
      char TempAdressR = 0b10000001; // Read adress

      unsigned int temp;
      PTBD_PTBD6=0;       //Sensor ON
      retardo(10);
      byteWrite(TempAdressW, 0xF3); // Measure Comand
      temp = byte2Read(TempAdressR);
      PTBD_PTBD6=1;       //Sensor OFF
      temp = (unsigned int)(((long)(temp*17572) >> 16) -4685);
    return(temp);
}



char leer_UV(void)
{
      char data1,data2, UV;
      unsigned int rawUV;
      PTBD_PTBD7=0;       //Sensor ON
      retardo(10);
      //Se implementa a bajo nivel porque no utiliza el estandar I2C.
```

```
       do  // first we send the address field
      {
      // enter master transmit mode and send a start
      IICC = bIICEN | bTX | bMST;
      // send ARA (acknowledge alert response)
      IICD = 0x18;
      while (!IICS_TCF);       // wait until the transmission ends
      } while (IICS_RXAK);     // if no ACK was received, resend the address field

      retardo(32000);          // wait 4s to measure
    do{
      IICD = 0x73;             // Ask for MSB
      while (!IICS_TCF);
    }while (IICS_RXAK);
    IICC_TX = 0;               // receive mode
    IICC_TXAK = 1;             // send NACK in the next read
    data1 = IICD;
    while (!IICS_TCF);         // wait until the transmission ends
    IICC_TX = 1;               // send mode
    do{
      IICD = 0x71;             // Ask for LSB
      while (!IICS_TCF);
    }while (IICS_RXAK);
    IICC_TX = 0;               // receive mode
    IICC_TXAK = 1;             // send NACK in the next read
    data2 = IICD;
    while (!IICS_TCF);         // wait until the transmission ends
    IICC = bIICEN;             // send a stop, leave master mode
    PTBD_PTBD7=1;              // Sensor OFF

    rawUV = (data1 << 8) | data2;
    if (rawUV <= 2988 ) UV = 0;
    else if ((rawUV > 2988) && (rawUV <= 5976)) UV = 1;
    else if ((rawUV > 5976) && (rawUV <= 8964)) UV = 2;
    else if ((rawUV > 8964) && (rawUV <= 11952)) UV = 3;
    else if ((rawUV > 11952) && (rawUV <= 14940)) UV = 4;
    else if ((rawUV > 14940) && (rawUV <= 17928)) UV = 5;
    else if ((rawUV > 17928) && (rawUV <= 20916)) UV = 6;
    else if ((rawUV > 20916) && (rawUV <= 23904)) UV = 7;
    else if ((rawUV > 23904) && (rawUV <= 26892)) UV = 8;
    else if ((rawUV > 26892) && (rawUV <= 29880)) UV = 9;
    else if ((rawUV > 29880) && (rawUV <= 32869)) UV = 10;
    else UV = 11;
    return(UV);
}
```

ad.h

```
/*****************************************************************
* Proyecto TFG
* Central meteorológica inalámbrica
*****************************************************************
* nombre fichero : ad.h
*****************************************************************
* descripcion    : Fichero de cabecera modulo conversor analogico -
digital
* programador    : Pablo Pastor
* lenguaje       : ANSI C para CodeWarrior
* fecha          : 8 julio 2017
*****************************************************************/
```

```
#ifndef ad_h
#define ad_h

void Init_AD (void) ;
unsigned int leer_LDR (void) ;
unsigned int leer_Battery (void);
char leer_Dir (void);
unsigned int leer_Vel (void);
void retardo(unsigned int tiempo);
#endif
```

ad.c

```
/*****************************************************************
* Proyecto TFG
* Central meteorológica inalámbrica
*****************************************************************
* nombre fichero : ad.c
*****************************************************************
* descripcion    : Implementacion modulo conversor analogico -digital
* programador    : Pablo Pastor
* lenguaje       : ANSI C para CodeWarrior
* fecha          : 8 julio 2017
*****************************************************************/


#include "derivative.h"
#include "ad.h"
int pulsos = 0;
interrupt 18 void InterrupcionVel(void);


void Init_AD (void) {
  ADCSC1_ADCO = 0;        // One conversion
  ADCSC1_AIEN = 0;        // Interrupt disabled

  ADCSC2_ADTRG = 0;       // Software trigger enabled
  ADCSC2_ACFE = 0;        // Compare function disabled

  ADCCFG_ADLPC = 0;       // High speed configuration
  ADCCFG_ADIV = 0;        // Input clock without division
  ADCCFG_ADLSMP = 1;      // Long sample time
  ADCCFG_MODE = 0;        // 8 bit conversion
  ADCCFG_ADICLK = 0;      // Bus clock source

  APCTL1_ADPC0 = 1;       // PTA0/ADP0 activation (LDR/PRESSURE)
  APCTL1_ADPC1 = 1;       // PTA1/ADP1 activation (BATTERY)
  APCTL1_ADPC6 = 1;       // PTB2/ADP6 activation (VEL/DIR)

}



unsigned int leer_LDR (void)
{
      char raw;
      unsigned int lux;
    PTBD_PTBD7=0;         // Activated sensor pwr-suply
    retardo(10);
    ADCSC1_ADCH = 0b00000;// AD conversion start
                          // Conversion chanel ADP0/PTA0
```

```
    while(!ADCSC1_COCO);  // Wait conversion end
    PTBD_PTBD7=1;         // Stop sensor pwr-suply
    raw = ADCR;           // Read data, Clean COCO
    if (raw <= 20 ) lux = 40000;
    else if ((raw > 20) && (raw <= 30)) lux = 16000;
    else if ((raw > 30) && (raw <= 40)) lux = 7700;
    else if ((raw > 40) && (raw <= 50)) lux = 4300;
    else if ((raw > 50) && (raw <= 60)) lux = 2600;
    else if ((raw > 60) && (raw <= 70)) lux = 1700;
    else if ((raw > 70) && (raw <= 80)) lux = 1100;
    else if ((raw > 80) && (raw <= 90)) lux = 770;
    else if ((raw > 90) && (raw <= 100)) lux = 540;
    else if ((raw > 100) && (raw <= 110)) lux = 380;
    else if ((raw > 110) && (raw <= 120)) lux = 270;
    else if ((raw > 120) && (raw <= 130)) lux = 195;
    else if ((raw > 130) && (raw <= 140)) lux = 140;
    else if ((raw > 140) && (raw <= 150)) lux = 100;
    else if ((raw > 150) && (raw <= 160)) lux = 70;
    else if ((raw > 160) && (raw <= 170)) lux = 50;
    else if ((raw > 170) && (raw <= 180)) lux = 30;
    else if ((raw > 180) && (raw <= 190)) lux = 20;
    else lux = 13;
    return lux;
}

unsigned int leer_Battery (void)
{
      char raw;
      unsigned int batt;
    PTBD_PTBD7=0;         // Activated sensor pwr-suply
    retardo(10);
    ADCSC1_ADCH = 0b00001;// AD conversion start
                          // Conversion chanel ADP1/PTA1
    while(!ADCSC1_COCO);  // Wait conversion end
    PTBD_PTBD7=1;         // Stop sensor pwr-suply
    raw = ADCR;               // Read data, Clean COCO
    batt = 2588 * raw;    // ADC configuration from 0-6.6V, returns V
*100 -> 3.83V = 383
    return batt;
}

char leer_Dir (void)
{
    PTBD_PTBD5=0;         // Activated sensor pwr-suply
    retardo(10);
    ADCSC1_ADCH = 0b00110;// AD conversion start
                          // Conversion chanel ADP6/PTB2
    while(!ADCSC1_COCO);  // Wait conversion end
    PTBD_PTBD5=1;         // Stop sensor pwr-suply
    return ADCR;          // Read data, Clean COCO
}

unsigned int leer_Vel (void)
{
      char raw,lap=0;
      unsigned int vel;
    PTBD_PTBD3=0;                   // Activated sensor pwr-suply
      pulsos =0;                        // Reset counter
    retardo(10);
    //KBI interrupt configuration
    KBIPE=0b01000000;               // enable KBIP0 at PTB2
```

```
    KBIES=0;                                // Edge select: edge 1 to 0
    KBISC=0x00000110;                   //INT enable, single edge detector
    //MTIM configuration
    MTIMCLK = 0b00001000;               //CLK source BUS clock, %256 --
>15625Hz
    MTIMMOD= 255;                       //Count 255, Reset timer
    MTIMSC_TOIE = 0;                    //No interrupt
    MTIMSC_TSTP = 0;                    //Start TIMER
    MTIMSC_TRST = 1;                    //Reset to 0
    for(lap=0; lap<=62; lap++){
      while(!MTIMSC_TOF);               //Wait to end lap
      MTIMSC_TRST = 1;                  //Reset to 0
    }
    MTIMSC_TSTP = 1;                    //Stop TIMER
    KBISC_KBIE=0;                       //INT disable
    PTBD_PTBD3=1;                       //Stop sensor pwr-suply
    vel = 22 * pulsos;
    return vel;
}

interrupt 18 void InterrupcionVel (void){
      pulsos+=1;
    KBISC_KBACK = 1;
}
```

sci.h

```
/*****************************************************************
* Proyecto TFG
* Central meteorológica inalámbrica
*****************************************************************
* nombre fichero : sci.h
*****************************************************************
* descripcion    : Fichero de cabecera modulo de comunicaciones sci
* programador    : Pablo Pastor
* lenguaje       : ANSI C para CodeWarrior
* fecha          : 28 agosto 2017
*****************************************************************/

#ifndef sci_h
#define sci_h
void  Init_SCI(void);
interrupt 15 void recibir (void);
void send_data(signed long P, unsigned int RH, unsigned int Temp, char
UV, char Irradiancia, char Dir, unsigned int Vel, unsigned int Batt);
#endif
```

sci.c

```
/*****************************************************************
* Proyecto TFG
* Central meteorológica inalámbrica
*****************************************************************
* nombre fichero : sci.c
*****************************************************************
* descripcion    : Implementacion modulo de comunicaciones SCI
* programador    : Pablo Pastor
* lenguaje       : ANSI C para CodeWarrior
* fecha          : 8 julio 2017
*****************************************************************/
```

```
#include "derivative.h"
#include "sci.h"

char ACK = 0;

void Init_SCI(void)
{
//Speed communication 9600b
  SCIBDH = 0;
  SCIBDL = 26;
//Enable Rx,Tx
  SCIC2_TE = 1;
  SCIC2_RE = 1;
  SCIC2_RIE=1; //Init interrupt
}

interrupt 15 void recibir (void)
{
    char aux=SCIS1_RDRF;
    ACK = SCID;
    SCIC2_ACK=1;
  }

send_data(signed long P, unsigned int RH, unsigned int Temp, char UV,
char Irradiancia, char Dir, unsigned int Vel, unsigned int Batt)
{
char aux;
PTBD_PTBD4=0;
//Send P
do{
      aux=(char)(P >> 24);
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
do{
      aux=(char)(P >> 16);
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
do{
      aux=(char)(P >> 8);
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
do{
      aux=(char)(P);
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
//Send RH
do{
      aux=(char)(RH >> 8);
      while(SCIS1_TDRE==0)
```

```
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
do{
      aux=(char)(RH);
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
//Send Temp
do{
      aux=(char)(Temp >> 8);
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
do{
      aux=(char)(Temp);
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
//Send UV
do{
      aux=UV;
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
//Send Irradiancia
do{
      aux=Irradiancia;
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
//Send Dir
do{
      aux=Dir;
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
//Send Vel
do{
      aux=(char)(Vel >> 8);
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
do{
      aux=(char)(Vel);
      while(SCIS1_TDRE==0)
```

```
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
//Send Batt
do{
      aux=(char)(Batt >> 8);
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
do{
      aux=(char)(Batt);
      while(SCIS1_TDRE==0)
      SCID=aux;
      retardo(1);
}while(ACK!='A');
ACK=0;
      retardo(25000); //Wait for server sync.
PTBD_PTBD4=1;

}
```

Anexo IX. Código implementado en el procesador esp8266

Se han implementado dos programaciones distintas. Para el ESP8266 como módulo wifi que se comunica con el procesador QG8 en los módulos externos y como unidad interior independiente.

En los módulos externos, utilizando la librería <*PubSubClient.h*> para implementar el protocolo MQTT:

```
/*
  Basic ESP8266 MQTT example, modified by Pablo.

  This sketch demonstrates the capabilities of the pubsub library in
combination
  with the ESP8266 board/library.

  It will reconnect to the server if the connection is lost using a
blocking
  reconnect function. See the 'mqtt_reconnect_nonblocking' example for
how to
  achieve the same result without blocking the main loop.

  To install the ESP8266 board, (using Arduino 1.6.4+):
  - Add the following 3rd party board manager under "File ->
Preferences -> Additional Boards Manager URLs":
       http://arduino.esp8266.com/stable/package_esp8266com_index.json
  - Open the "Tools -> Board -> Board Manager" and click install for
the ESP8266"
  - Select your ESP8266 in "Tools -> Board"

*/

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <string.h>

// Update these with values suitable for your network.
const char* ssid = "TP_TFG";
const char* password = "tfg_pruebas";
const char* mqtt_server = "brokerTFG";

WiFiClient espClient;
PubSubClient client(espClient);

char data1, data2, data3, data4;
char msg[50];
signed long P;
unsigned int RH, Temp, Vel, Batt;
char UV, Dir;
float floP,floTemp,floVel,

void setup() {
  Serial.begin(9600); //Comunication with QG8
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
```

```
void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    // Attempt to connect
    if (client.connect("ESP8266Client")) {
    } else {
      // Wait 500 miliseconds before retrying
      delay(500);
    }
  }
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  //Read P
  data1=Serial.read();
  Serial.print("A");
  data2=Serial.read();
  Serial.print("A");
  data3=Serial.read();
  Serial.print("A");
  data4=Serial.read();
  Serial.print("A");
  P=(signed long)(data1<<24)+(signed long)(data2<<16)+(signed
long)(data3<<8)+(data4);
  floP=P/100;
  client.publish("casa/TFG/presion", floP);
  //Read RH
  data1=Serial.read();
  Serial.print("A");
  data2=Serial.read();
  Serial.print("A");
  RH=(unsigned int)(data1<<8)+(data2);
  client.publish("casa/TFG/humedad", RH);
  //Read Temp
  data1=Serial.read();
  Serial.print("A");
  data2=Serial.read();
  Serial.print("A");
  Temp=(unsigned int)(data1<<8)+(data2);
  floTemp=Temp/100;
  client.publish("casa/TFG/temperatura", floTemp);
  //Read UV
  data1=Serial.read();
  Serial.print("A");
  client.publish("casa/TFG/UV", data1);
  //Read Irradiancia
  data1=Serial.read();
```

```
  Serial.print("A");
  client.publish("casa/TFG/Irradiancia", data1);
  //Read Dir
  data1=Serial.read();
  Serial.print("A");
  client.publish("casa/TFG/DireccionViento", data1);
  //Read Vel
  data1=Serial.read();
  Serial.print("A");
  data2=Serial.read();
  Serial.print("A");
  Vel=(unsigned int)(data1<<8)+(data2);
  floVel=Vel/100;
  client.publish("casa/TFG/VelocidadViento", floVel);
  //Read Batt
  data1=Serial.read();
  Serial.print("A");
  data2=Serial.read();
  Serial.print("A");
  Batt=(unsigned int)(data1<<8)+(data2);
  floBatt=Batt/100;
  client.publish("casa/TFG/Bateria", floBatt);
  }
}
```

Para los módulos internos se configura de igual manera el protocolo MQTT y además la conexión I2C con el sensor de temperatura y humedad:

```
/*
 * Wire - I2C Scanner
 *
 * The WeMos D1 Mini I2C bus uses pins:
 * D1 = SCL
 * D2 = SDA
 */

#include <Wire.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "TP_TFG";
const char* password = "tfg_pruebas";
const char* mqtt_server = "brokerTFG";

const int sclPin = D1;
const int sdaPin = D2;
char data1, data2;
unsigned int rawT, rawRH;
float t, RH;

WiFiClient espClient;
PubSubClient client(espClient);

void setup()
{
  Wire.begin(sdaPin, sclPin);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
```

```
void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    // Attempt to connect
    if (client.connect("ESP8266Client")) {
    } else {
      // Wait 500 miliseconds before retrying
      delay(500);
    }
  }
}

void loop() {
      if (!client.connected()) {
    reconnect();
  }
  client.loop();

  //Read RH
  Wire.beginTransmission(0b10000000); // write mode
  Wire.write(byte(0xF5));             // sends measurement petition
  Wire.endTransmission();            // stop transmitting
  while (!Wire.available());          // wait conversion
  Wire.beginTransmission(0b10000001); // read mode
  data1=Wire.read();
  while (!Wire.available());
  data2=Wire.read();
  Wire.endTransmission();            // stop transmitting
  rawRH = (data1 << 8) | data2
  RH = (rawRH*125)/65536-6;

  //Read Temp
  Wire.beginTransmission(0b10000000); // write mode
  Wire.write(byte(0xF3));             // sends measurement petition
  Wire.endTransmission();            // stop transmitting
  while (!Wire.available());          // wait conversion
  Wire.beginTransmission(0b10000001); // read mode
  data1=Wire.read();
  while (!Wire.available());
  data2=Wire.read();
  Wire.endTransmission();            // stop transmitting
  rawT = (data1 << 8) | data2
  t = (rawT*175.72)/65536-46.85;
  client.publish("casa/TFG/humedad_int", RH);
  client.publish("casa/TFG/temperatura_int", t);
  for(int temp = 0; temp <= 300; temp++){  //Blocking code for 5min
  delay(1000);
  }
}
```

Anexo X. Configuración servidor central

INSTALANDO SERVIDOR IOT en RASPBERRY PI Zero W

-Descargar distro Raspbian Lite
(https://www.raspberrypi.org/downloads/raspbian/)
-Instalarlo con Rufus (https://rufus.akeo.ie/?locale)

-Crear archivo wpa_supplicant.conf con:

```
country=ES
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
        ssid="TP_TFG"
        psk="tfg_pruebas"
        key_mgmt=WPA-PSK
}
```

-Crear archivo ssh (sin extensión) para activar la conexión remota

-Introducir tarjeta SD en la raspberry y acceder mediante putty (user: pi pswrd: raspberry).

-Cambiar contraseña con passwd
-Para establecer una IP fija:

```
sudo nano /etc/dhcpcd.conf
```

añadir:

```
interface wlan0
static ip_address=192.168.1.200
static routers=192.168.1.1
static domain_name_servers=8.8.8.8
static domain_search=8.8.4.4
```

- Ir a sudo nano /etc/network/interfaces y añadir

```
auto lo
iface lo inet loopback
iface wlan0 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

-Configuración básica lista.

(https://community.openhab.org/t/influxdb-grafana-persistence-and-graphing/13761/21)
(https://community.openhab.org/t/influxdb-grafana-persistence-and-graphing/13761/88)

INSTALACIÓN BÁSICA Grafana

- sudo apt-get install apt-transport-https curl
- curl https://bintray.com/user/downloadSubjectPublicKey?username=bintray | sudo apt-key add -
- echo "deb https://dl.bintray.com/fg2it/deb-rpi-1b jessie main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
- sudo apt-get update
- sudo apt-get install grafana
- sudo service grafana-server start
- sudo update-rc.d grafana-server defaults

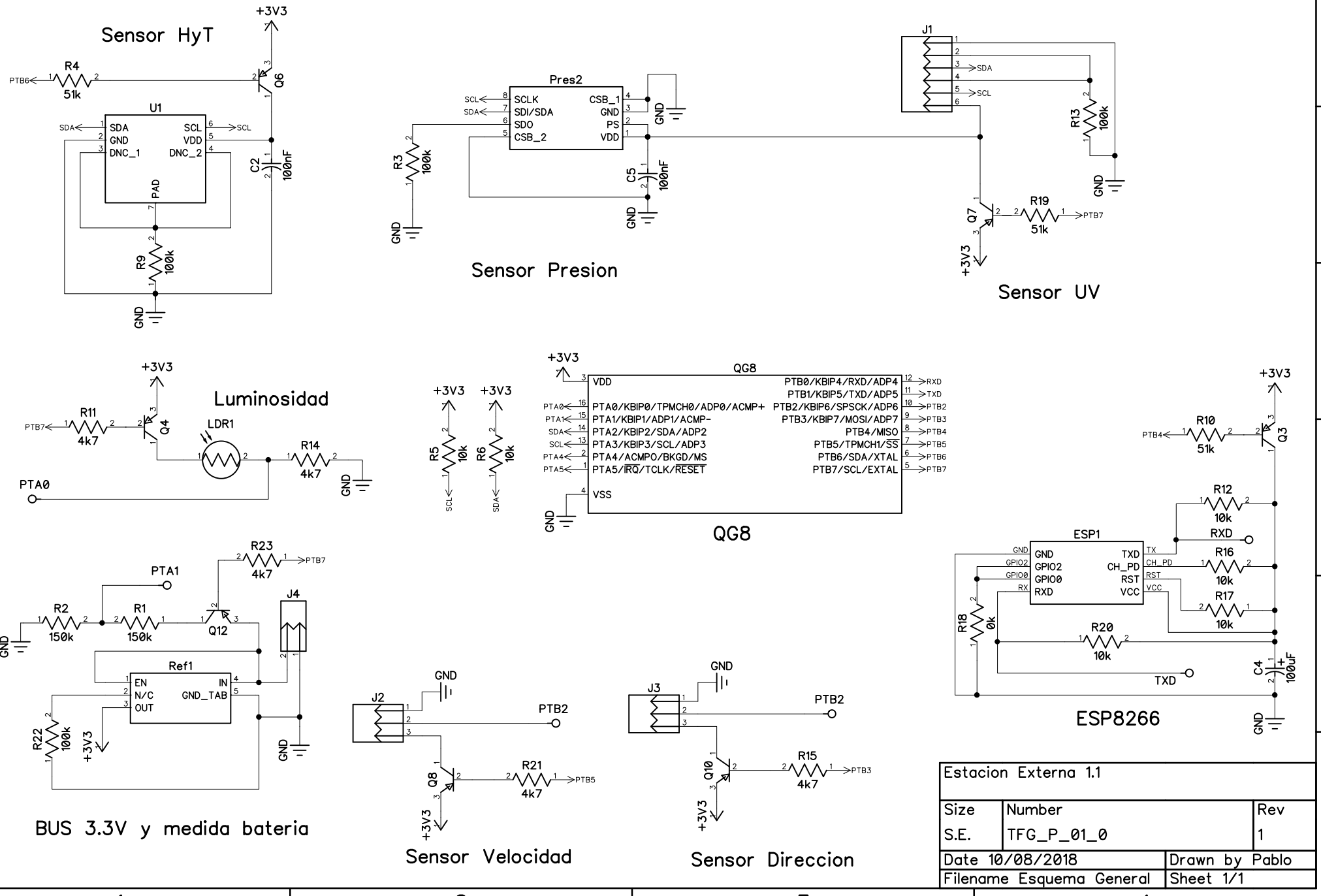
INSTALACIÓN BÁSICA OPENHAB 2
(https://www.openhab.org/docs/installation/rasppi.html)

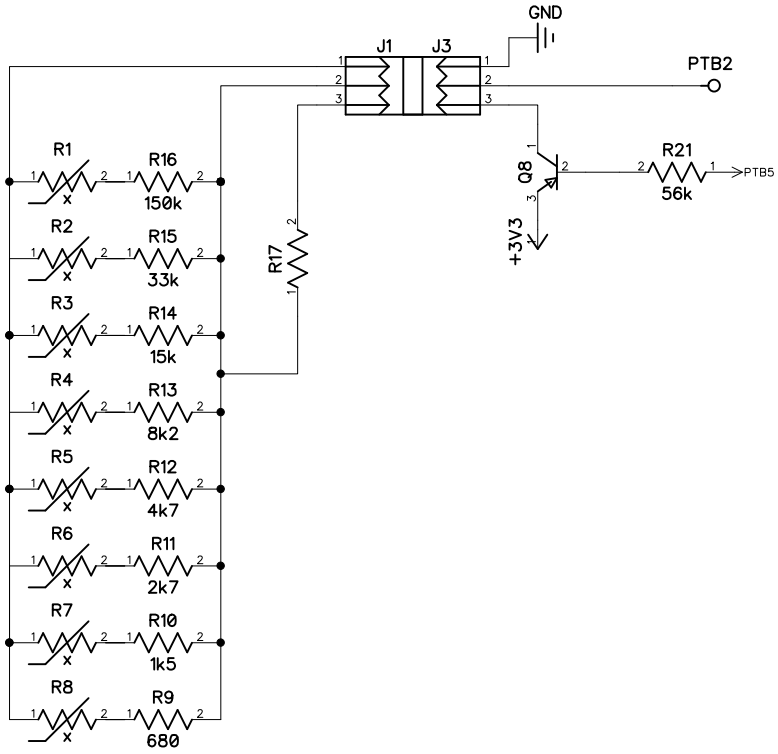
- sudo apt-get install oracle-java8-jdk
- sudo apt-get update
- sudo apt-get upgrade
- wget -qO - 'https://bintray.com/user/downloadSubjectPublicKey?username=openhab' | sudo apt-key add -
- sudo apt-get install apt-transport-https
- echo 'deb https://dl.bintray.com/openhab/apt-repo2 stable main' | sudo tee /etc/apt/sources.list.d/openhab2.list
- sudo apt-get update
- sudo apt-get install openhab2
- sudo /etc/init.d/openhab2 start
- sudo /etc/init.d/openhab2 status
- sudo update-rc.d openhab2 defaults

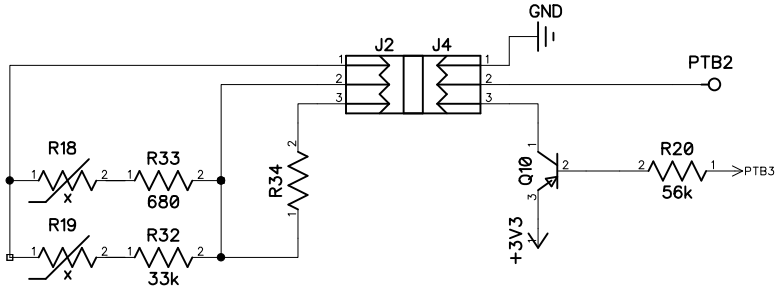
INSTALACIÓN BÁSICA InfluxDB

- wget -O - https://repos.influxdata.com/influxdb.key | sudo apt-key add -
- echo "deb https://repos.influxdata.com/debian jessie stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
- sudo apt-get update
- sudo apt-get install influxdb
- sudo systemctl daemon-reload
- sudo systemctl enable influxdb.service
- sudo systemctl start influxdb.service

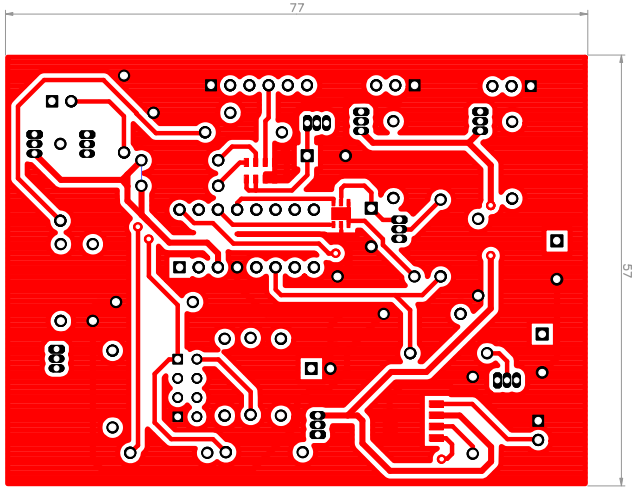
Anexo XI. Planos PCB diseñadas


Sensor HyT
+3V3
R4
51k
PTB6
Q6
U1
SDA
GND
DNC_1
SCL
VDD
DNC_2
PAD
C2
100nF
R9
100k
GND
Pres2
SCLK
SDI/SDA
SDO
CSB_2
CSB_1
GND
PS
VDD
R3
100k
C5
100nF
Sensor Presion
J1
SDA
SCL
R13
100k
Q7
R19
51k
PTB7
Sensor UV
Luminosidad
R11
4k7
Q4
LDR1
R14
4k7
PTA0
R5
10k
R6
10k
QG8
VDD
PTA0/KBIP0/TPMCH0/ADP0/ACMP+
PTA1/KBIP1/ADP1/ACMP-
PTA2/KBIP2/SDA/ADP2
PTA3/KBIP3/SCL/ADP3
PTA4/ACMPO/BKGD/MS
PTA5/IRQ/TCLK/RESET
VSS
PTB0/KBIP4/RXD/ADP4
PTB1/KBIP5/TXD/ADP5
PTB2/KBIP6/SPSCK/ADP6
PTB3/KBIP7/MOSI/ADP7
PTB4/MISO
PTB5/TPMCH1/SS
PTB6/SDA/XTAL
PTB7/SCL/EXTAL
QG8
R10
51k
Q3
R12
10k
RXD
ESP1
GND
GPIO2
GPIO0
RX
TXD
CH_PD
RST
VCC
R16
10k
R17
10k
R18
0k
R20
10k
C4
100uF
ESP8266
R23
4k7
PTA1
R2
150k
R1
150k
Q12
J4
Ref1
EN
N/C
OUT
IN
GND_TAB
R22
100k
BUS 3.3V y medida bateria
J2
PTB2
Q8
R21
4k7
PTB5
Sensor Velocidad
J3
Q10
R15
4k7
PTB3
Sensor Direccion
Estacion Externa 1.1
Size
S.E.
Number
TFG_P_01_0
Rev
1
Date 10/08/2018
Drawn by Pablo
Filename Esquema General
Sheet 1/1

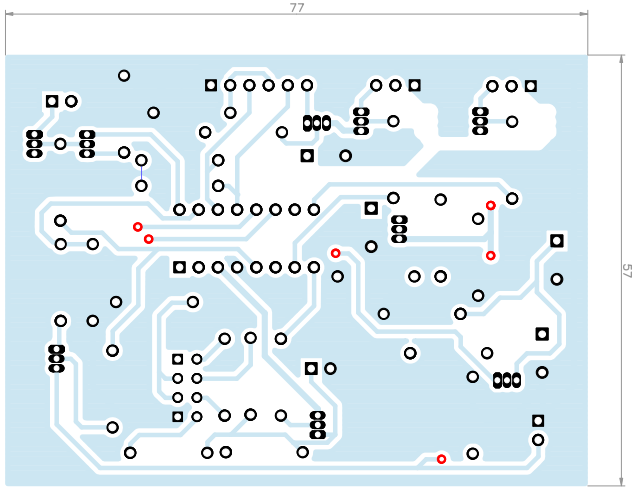




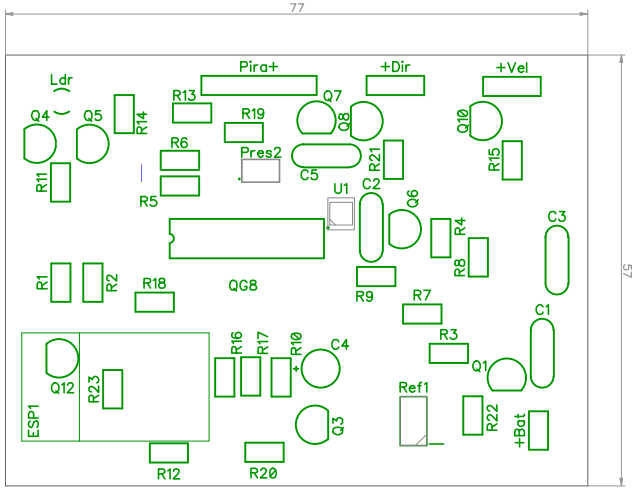
Esquema Veleta y Anemometro		
Size S.E.	Number TFG_P_02_1	Rev 1
Date 12/08/2018		Drawn by Pablo
Anemometro		Sheet 1/1



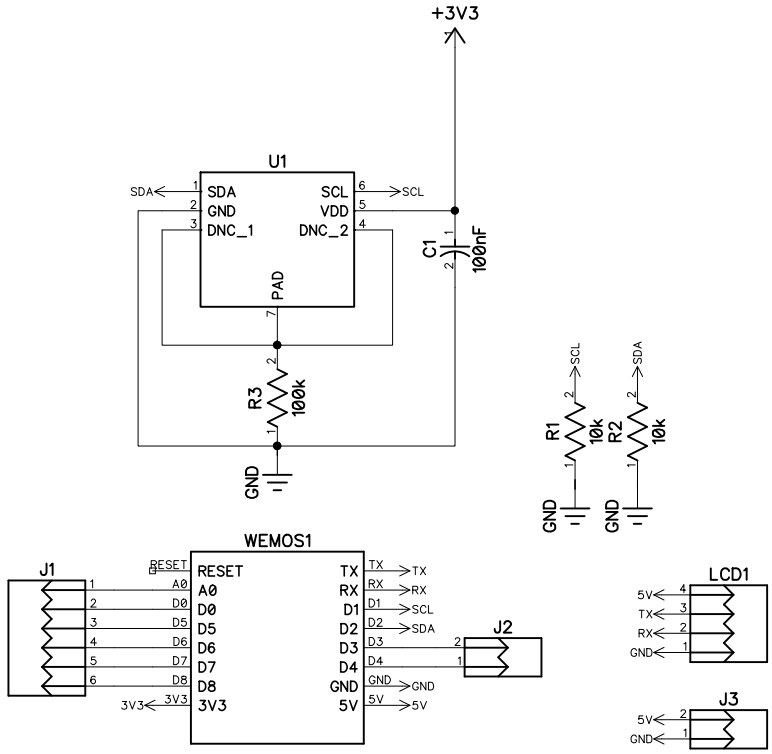
	Fecha:	Nombre:		
Dibujado:	08-18	**Pablo Pastor**		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Escala: 1:1	Título: PCB_externa_top			
	Proyecto: TFG			Curso: 4º Electrónica
				Nº Plano: 3



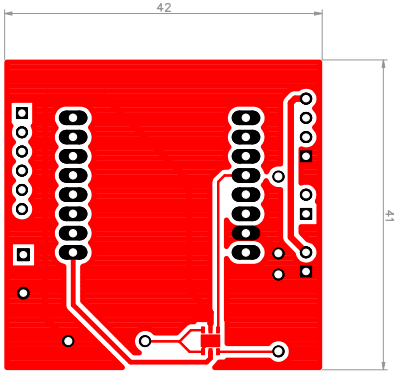
	Fecha:	Nombre:		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	**Pablo Pastor**		
Escala: 1:1	Título: PCB_externa_bottom			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: 1



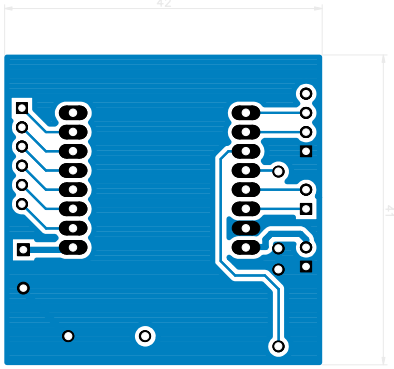
	Fecha:	Nombre:		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	**Pablo Pastor**		
Escala: 1:1	Título: PCB_externa_serigrafia			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: 2



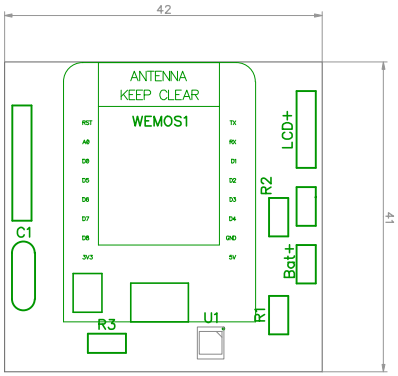
Estacion interior 1.1		
Size S.E.	Number TFG_P_01_1	Rev 1
Date 10/08/2018		Drawn by Pablo
Filename Esquema General		Sheet 1/1



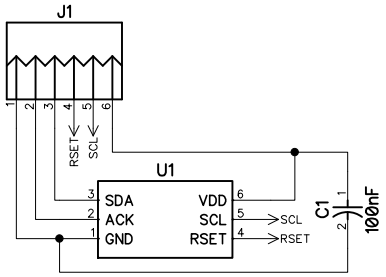
	Fecha:	Nombre:		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	**Pablo Pastor**		
Escala: 1:1	Título: PCB_interna_top			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: 6



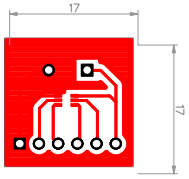
	Fecha:	Nombre:		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	**Pablo Pastor**		
Escala: 1:1	Título: PCB_interna_bottom			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: 4



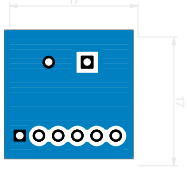
	Fecha:	Nombre:		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	**Pablo Pastor**		
Escala: 1:1	Título: PCB_interna_serigrafia			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: 5



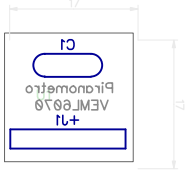
Esquema Piranometro		
Size S.E.	Number TFG_P_02_0	Rev 1
Date 27/07/2018		Drawn by Pablo
Esquema Piranometro		Sheet 1/1



	Fecha:	Nombre:		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	**Pablo Pastor**		
Escala: 1:1	Título: PCB_UV_top			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: 9



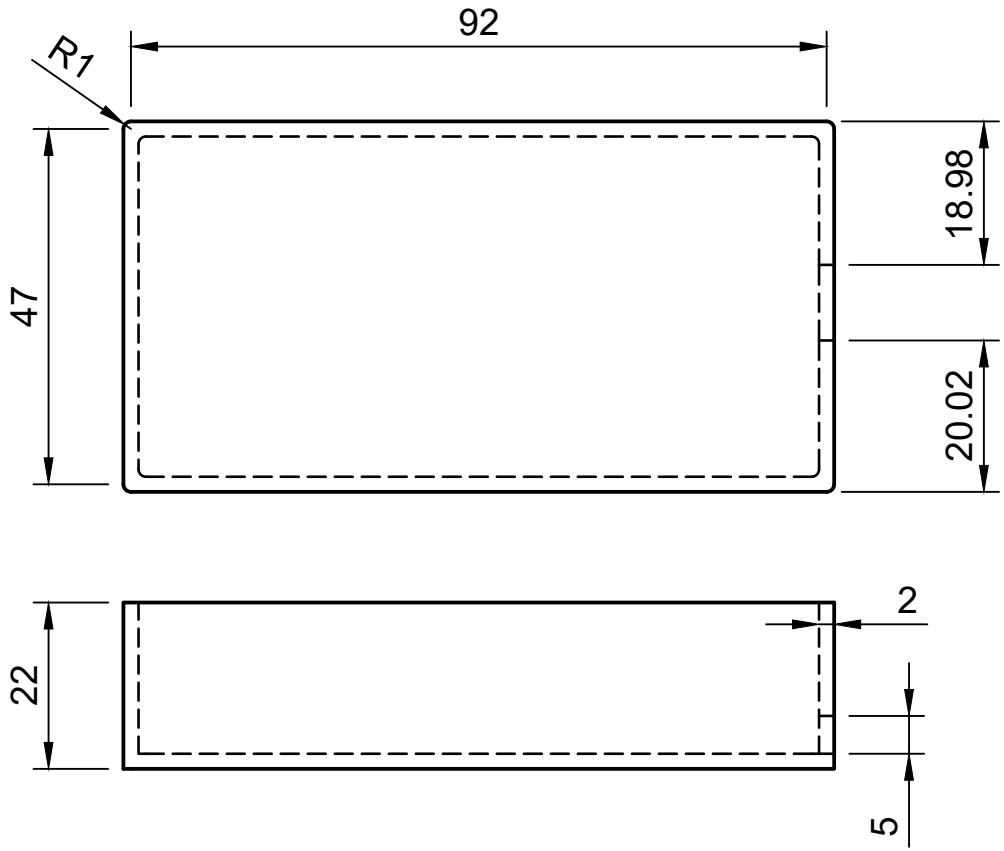
	Fecha:	*Nombre:*		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	***Pablo Pastor***		
Escala: 1:1	*Título:* PCB_UV_bottom			
	Proyecto: TFG			*Curso: 4º Electrónica*
				Nº Plano: 7

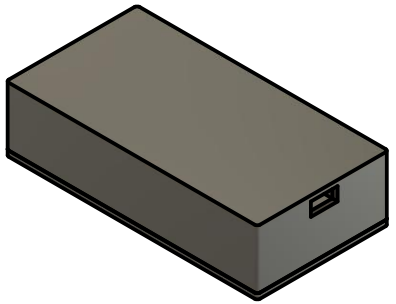


	Fecha:	Nombre:		
Dibujado:	08-18	**Pablo Pastor**		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Escala: 1:1	Título: PCB_UV_serigrafia			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: 8

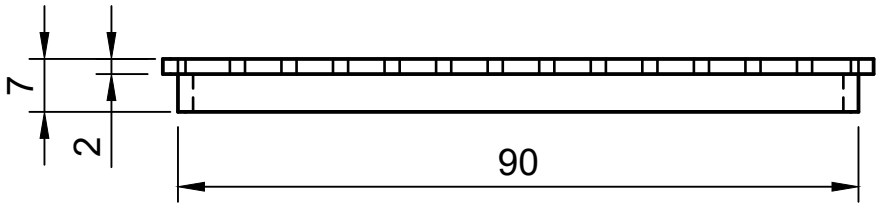
Anexo XII. Planos Carcasas

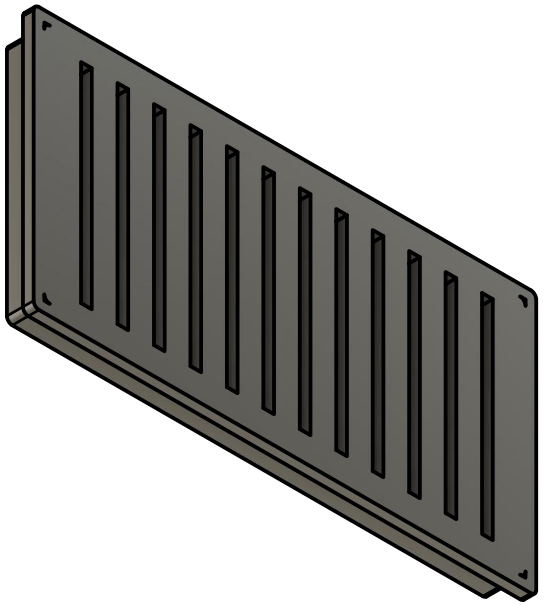
*La carcasa del módulo interior empotrable ha sido adaptada desde el diseño: https://www.thingiverse.com/thing:2603232

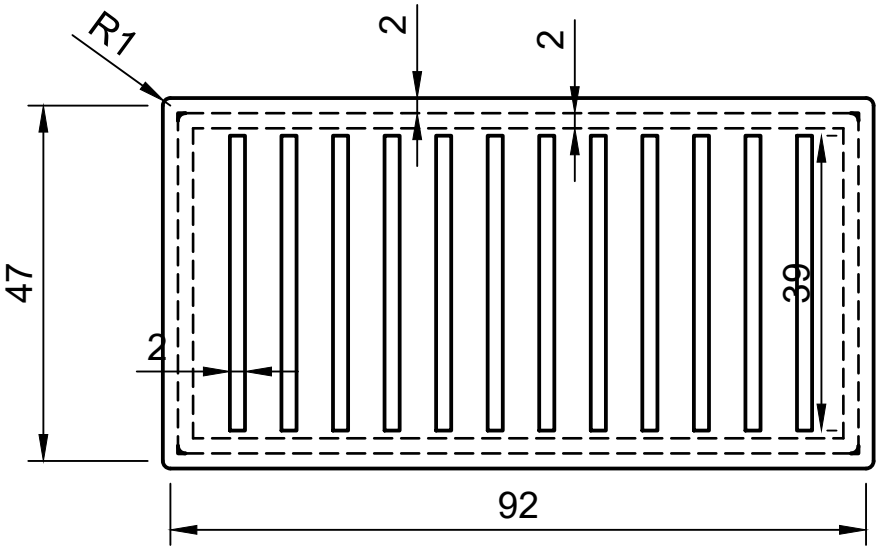




	Fecha:	Nombre:		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	**Pablo Pastor**		
Escala: 1:1	Título: Caja			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: Caja 1

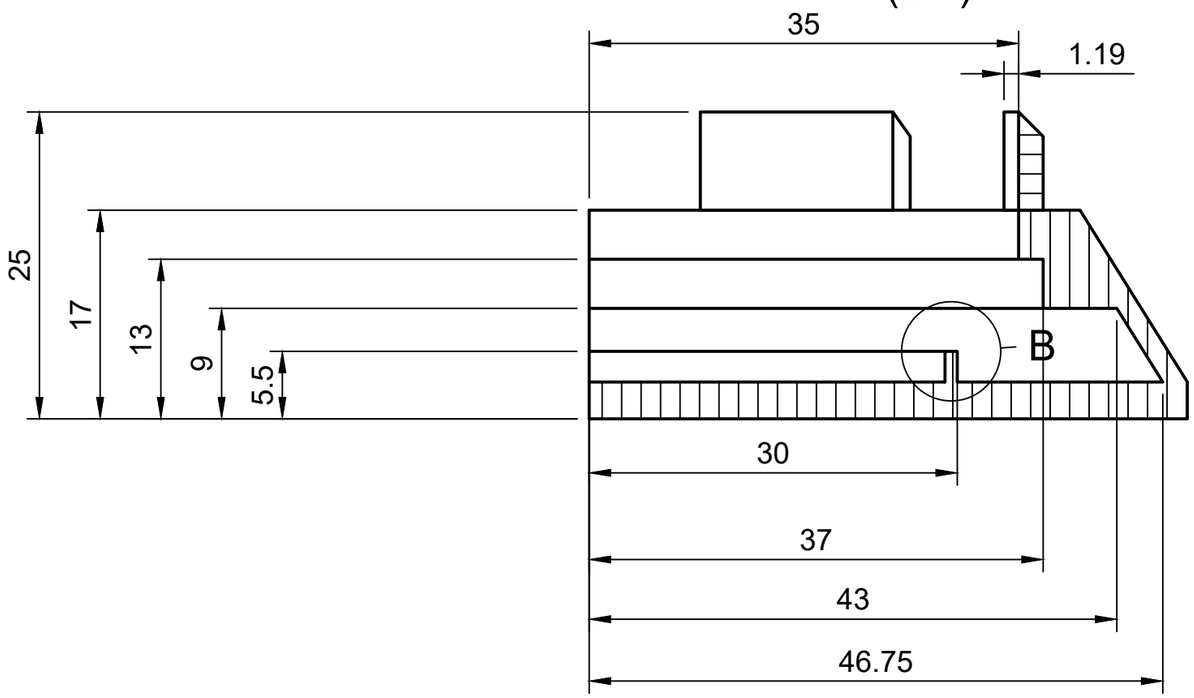




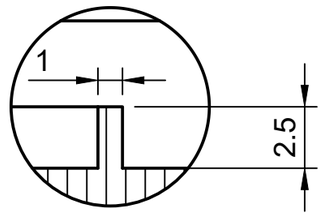


	Fecha:	Nombre:		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	**Pablo Pastor**		
Escala: 1:1	Título: Tapa caja			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: Caja 2

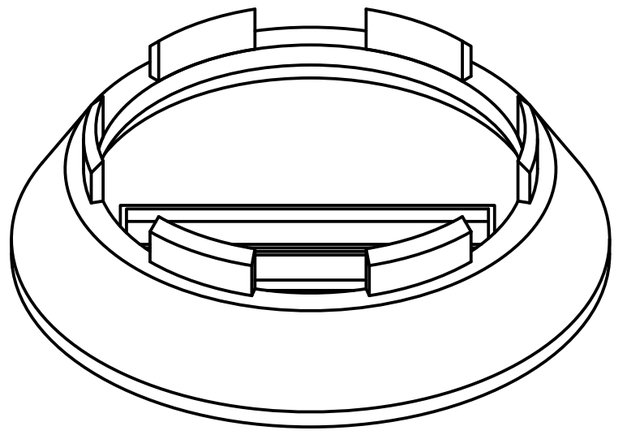
A-A (2:1)

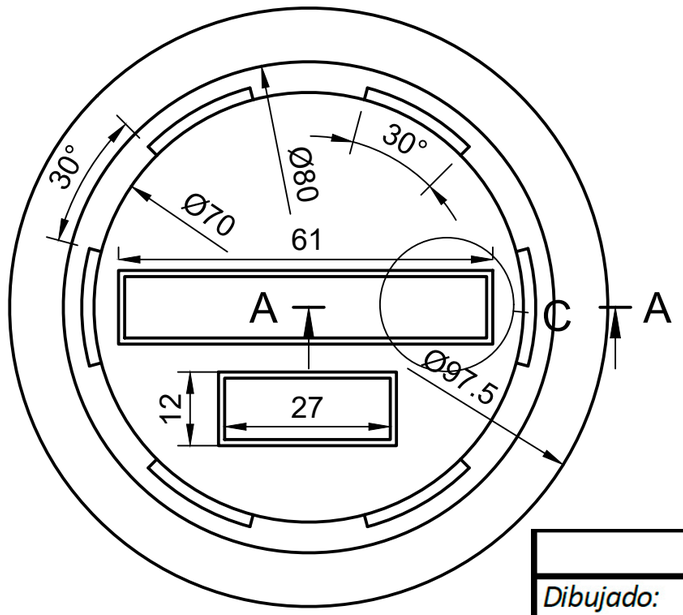


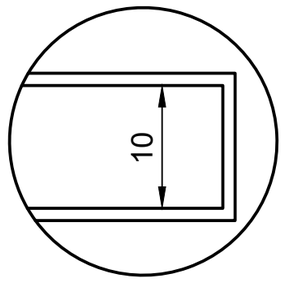
B (4:1)



C (2:1)

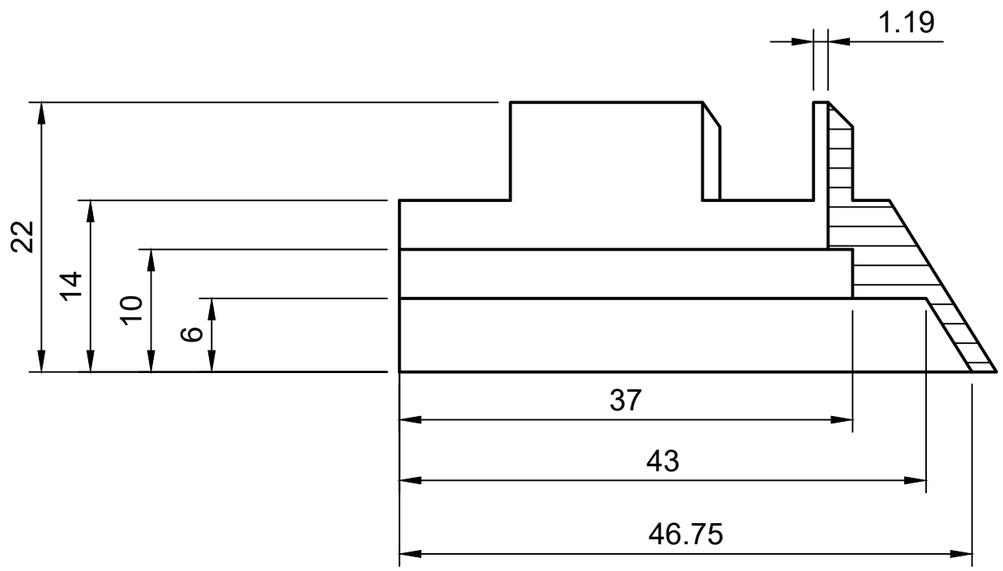


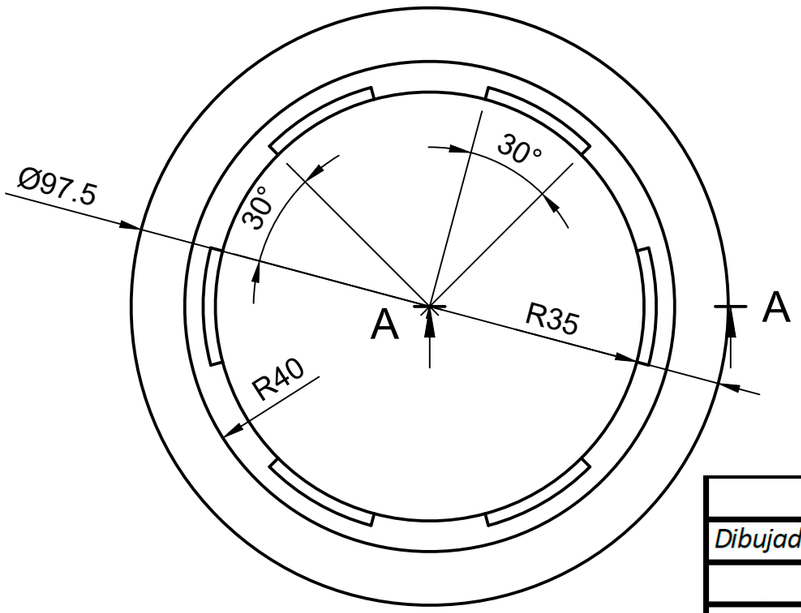




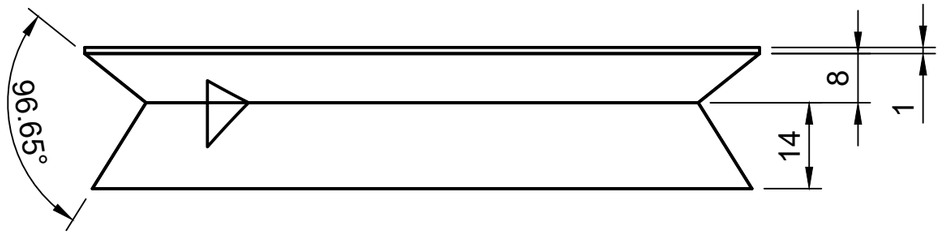
	Fecha:	Nombre:		
Dibujado:	08-18	**Pablo Pastor**		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Escala: 1:1	Título: Stevenson Inferior			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: Stevenson 1

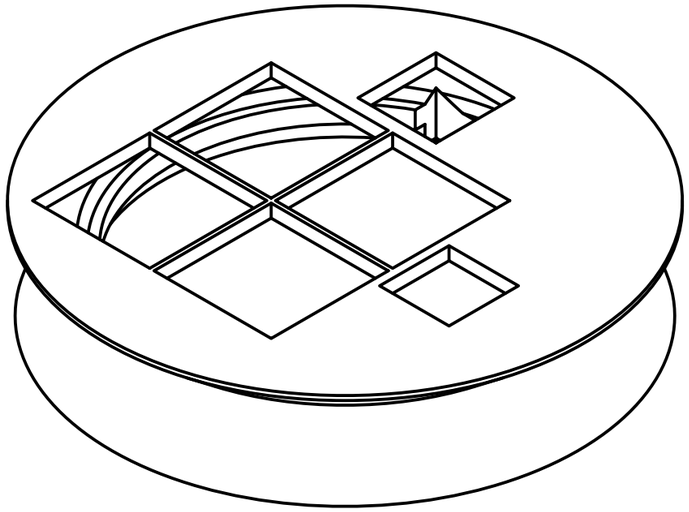
A-A (2:1)



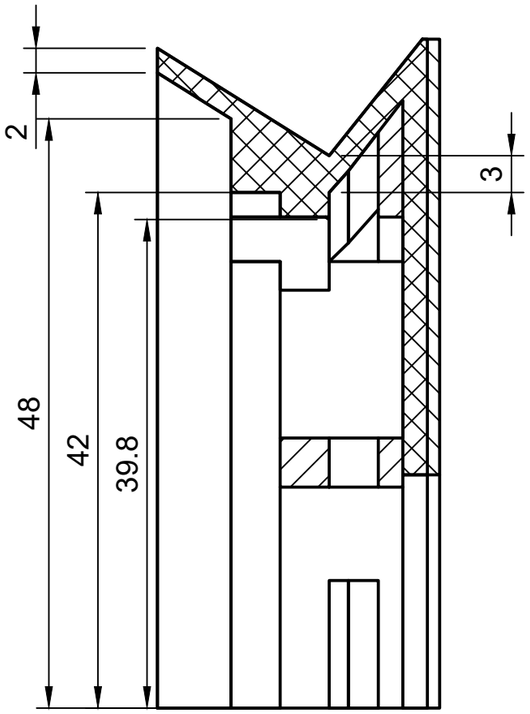


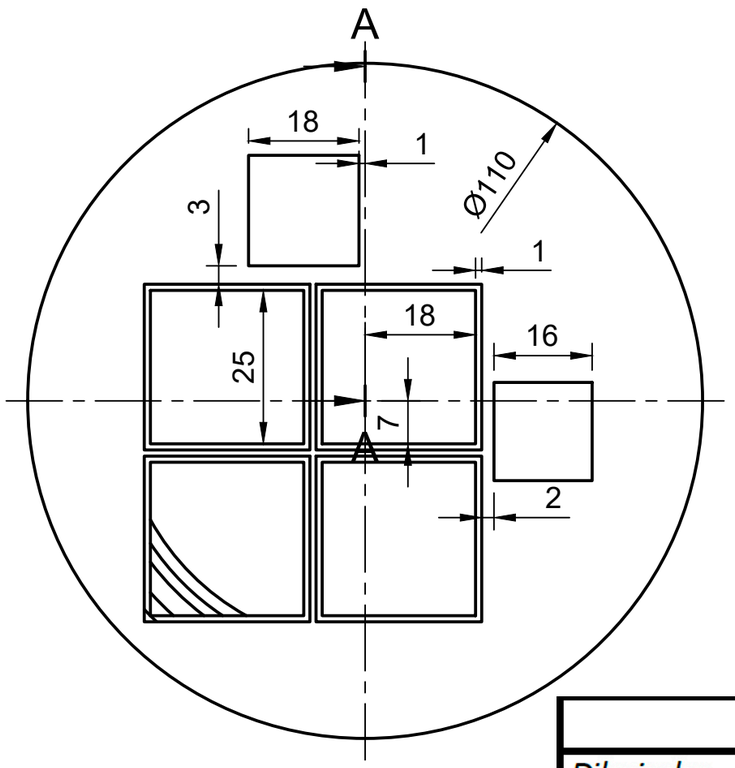
	Fecha:	Nombre:		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	**Pablo Pastor**		
Escala: 1:1	Título: Stevenson Lamas			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: Stevenson 2





A-A (2:1)





	Fecha:	Nombre:		Escuela de Ingeniería y Arquitectura Universidad Zaragoza
Dibujado:	08-18	**Pablo Pastor**		
Escala: 1:1	Título: Stevenson Superior			Curso: 4º Electrónica
	Proyecto: TFG			Nº Plano: Stevenson 3

Anexo XIII. Presupuesto

Se presenta las cantidades y coste de los elementos utilizados en cada uno de los nodos y el servidor central.

Nodo exterior	Nombre	Fabricante	Uds	Precio	Total
Sensor UV	VEML6070	Vishay	1	2,18 €	2,18 €
Sensor T. y H.	Si7006-A20-IM	Silicon Labs	1	1,67 €	1,67 €
Sensor presión	MS560702BA03-50	TE Connectivity	1	3,01 €	3,01 €
Reed Switch	-	-	10	0,68 €	6,80 €
Referencia de tensión	LP38692MP	Texas Instuments	1	0,94 €	0,94 €
Microprocesador	HCS08QG8	NXP	1	2,88 €	2,88 €
Módulo Wifi	ESP8266	Espressif	1	1,73 €	1,73 €
LDR	-	Luna Optoelectronics	1	0,45 €	0,45 €
Batería 600mAh	-	FamilyMall	1	2,40 €	2,40 €
Célula solar	Baliza solar	Tobago Led	4	1,79 €	7,16 €
Cargador batería	TP4056	MCIGICM	1	1,39 €	1,39 €
Carcasa 3D	-	-	1	1,68 €	1,68 €
PCB	-	-	1	1,25 €	1,25 €
				Total:	33,53 €

Nodo interior	Nombre	Fabricante	Uds	Precio	Total
Microprocesador	ESP8266 Mini NodeMCU	Espressif	1	2,36 €	2,36 €
Sensor T. y H.	Si7006-A20-IM	Silicon Labs	1	1,67 €	1,67 €
Fuente de alimentación		-	1	0,82 €	0,82 €
PCB	-	-	1	1,25 €	1,25 €
Carcasa 3D	-	-	1	0,59 €	0,59 €
				Total:	6,69 €

Servidor Central	Nombre	Fabricante	Uds	Precio	Total
Raspberry Pi Zero W		-	1	10,44 €	10,44 €
Carcasa	-	-	1	0,99 €	0,99 €
				Total:	11,43 €

Coste total del proyecto:	51,65 €

No se ha incluido el coste de componentes pasivos o adicionales.

Anexo XIV. Diagrama de Gantt

Nombre	Fecha de ini...	Fecha de fin
Revisión IoT	3/07/17	28/02/18
Protocolos I2C	3/07/17	2/10/17
Revisión Anemómetros	3/07/17	14/09/17
Revisión modelos térmicos	1/02/18	30/04/18
Revisión ESP8266	1/12/17	31/01/18
Revisión MQTT	1/03/18	30/03/18
Desarrollo anemómetro/veleta	9/07/18	20/07/18
Prueba de servicios web	30/04/18	31/05/18
Selección componentes	2/07/18	5/07/18
Diseño PCB	1/08/18	31/08/18
Desarrollo código	6/07/18	31/08/18
Diseño carcasas	30/08/18	7/09/18
Escritura memoria	3/09/18	14/09/18