TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

All the praises for Allah Almighty, who gave us the ability of hard work and courage as an ultimate consequence of which we became able to complete the project at hand with the required goals and much before the prescribed limit of time factor.

Secondarily, we, the associate workers of the project under study, are thankful to our project supervisor **Dr. Falah Mohammad**, through the kind guidance of which we were able to complete the project. He is absolutely a legend in the field of Telecommunication Engineering. In spite of his job, he arranged a number of meetings with us which proved to be very useful on our part. Sometimes, one short meeting with him helped solve the problems which might have taken days if we tried them on our own.

In the end, we consider it ultimate to pay regards to our parents and all the teachers of the Telecommunication Engineering Department, from which we learnt a lot throughout our 5-year course of study. It was not just the matter of final year, except the required competitive aptitude, sense of responsibility and sincerity required for the successful completion of any project was developed in us by our graceful parents and teachers during our 5-year period in the college.

ABSTRACT

In this project we aim to design weather station in order to provide us with some weather parameters every day such as temperature, humidity, daily amount of rainfall level, barometric pressure and wind speed. The main function for this station is to send data two times or more during a day by using the global System for Mobile communication (GSM) modem which is connected directly with a microcontroller. To make the project more efficient, some parameters are taken into account like cost, availability of components, the project time life, consumption power, accurately at sending and receiving of data.

Chapter 1

Introduction

1.1 Background of the Project

1.2 Objectives

CHAPTER 1 Introduction

1.1 Background of the Project

A wireless weather station is a standalone system that keeps the status of the outdoor temperature, humidity, daily amount of rainfall, barometric pressure, and wind speed; which is an outdoor weather monitoring system that gives easy way to keep the status of the outside weather. It has helped to travel safer without getting caught in bad weather condition like rain, snow, and fog. Wireless weather stations keeps us equipped with the weather report whenever and wherever. They are meant to give moment by moment information about changing of weather conditions. The main idea of this project is collecting information for weather conditions. Wireless weather substation being useful and portable.

Weather forecasts is the most important services, which are used by government to protect life and property and to improve the efficiency of operations, and by individuals to plan a wide range of daily activities, today weather forecast is a highly developed that is based in scientific principle and method. The name of this project is Mobile Weather Forecast station, that means a mobile will be a meteorological station, so that mobile will receive measurements for some weather parameters as temperature, humidity, barometric pressure, wind speed and the daily amount of rainfall, mobile will receive all measurements by using GSM modem (the Global System for Mobile Communication) that will send data to it, so that it is very important idea that is grounded on the GSM modem which is very important in weather forecasts.

1.2 objectives:

- To design and implement a mini weather station that can read the temperature, humidity, amount of daily rainfall, barometric pressure, and wind speed at the deployed point.
- To investigate the optimum method to report the data including utilizing the global System for Mobile communication (GSM).

CHAPTER 2

Overview of Design and Components

2.1 Review of Similar Systems

2.1.1 The Weather station

2.1.2 Home Made Logger of Temperature and Solar Data

2.1.3 Tenerife Weather Station

2.1.4 Weather Station (EasiData Mark 4)

2.2 Overview of project design

2.3 Review of Components

CHAPTER 2

Overview of Design and Components

This chapter contains three sections, first section is an overview of similar system, second is an overview of the project design, and the last section will demonstrate all components that are used in the overview of this design.

2.1 Review of Similar Systems

2.1.1The Weather station

This project was for students in Houston Baptist University it measures temperature, rainfall, humidity, barometric pressure and wind speed, the aim of this project is to compare data gathered from the weather station in 2011 to local weather since 1900, and in the near future include working with Information Technology Services to make up-to-date campus weather conditions available to all University students through portal. Additional plans include installing a lightning rod that will be beneficial for outdoor sporting events and a soil density reader for biology lab, this project shown as figure 2.1.1.1. The Project does not contain GSM modem to provide the destination by collected data moment by moment [1].

Figure 2.1.1.1: The Weather Station Project

2.1.2 Home Made Logger of Temperature and Solar Data:

This project made by David Cook, it is designed to be located away from the house without a power cord and it is also Operate for long periods of time without human intervention or maintenance, retain data in the event of a power loss or microcontroller reset. Survive all outdoor temperatures and conditions (such as rain and snow).The weather station consists of a small solar panel (about 1/8 watt), rechargeable battery backup for night time and cloudy operation, microcontroller ,data stored in a 4 MB flash chip, two power source monitors, and six temperature sensors. The six temperature sensors four are taped to a bamboo pole at various heights, one is in the project box, and one is a couple of inches underground. This project not has GSM for giving data moment by moment, and don't have the ability to provide the other conditions of the weather like wind and direction, and humidity, this project shown as Figure 2.1.2.1.

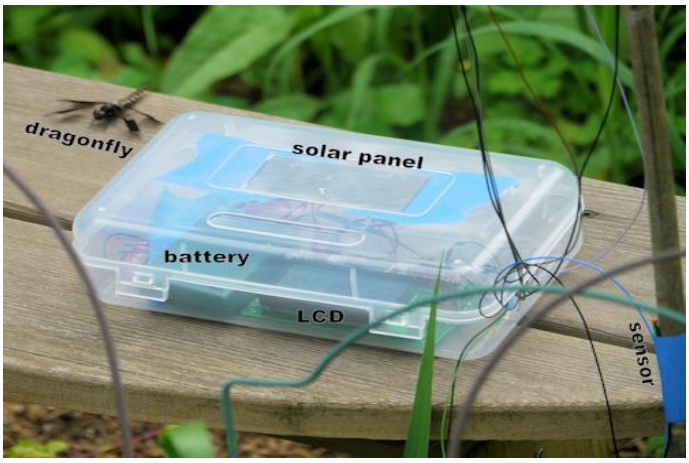


Figure 2.1.2.1: Home Made Logger of Temperature and Solar Data

2.1.3 Tenerife Weather Station

The weather station runs 24 hours a day, 7 days a week and 365 days a year; logging data and providing weather status information to many systems within the observatory, The weather station consist of many sensors, wind speed 1, wind speed 2, wind direction, internal temperature, external temperature, rain, dew, cloud south north, east, and west, solar radiation, light sun rise and set, barometric pressure. One function of the weather station is to provide a good or bad signal to the control server, the good or bad signal is generated every 10 seconds; the output is controlled by a series of rules, if any of the rules report that the weather is bad the output signal is set to bad, this information is used to govern the operation of the control server. In addition to the control server the good or bad output is also used by numerous safety systems that act to protect the system in the event of a failure elsewhere in the system. The weather station collects samples from each sensor every 10 seconds; every 10 minutes the weather station calculates average, minimum and maximum values and inserts them into the database, from here the database logs are uploaded to the primary UK server, from which they are used to generate the weather statistics and graphs. This project needs to connect directly to

a source power, so this project not useful to be installed in isolated area to measured weather conditions, and also it is more cost.

Figure 2.1.3.1 Tenerife Weather Station

2.1.4 Weather Station (EasiData Mark 4)

The EasiData Mark 4 is a highly flexible data logger that until 2009 formed the core component of Environdata's modular weather stations. These weather stations can be customized to suit almost any application or location, The EasiData Mark 4 can store up to 216 data types, including real time calculations on sensor inputs (eg. Current sensor readings, means, maximums, minimums, true vector wind analysis, evaporation rate etc). New commands can be sent to the logger at any time to reconfigure its storage operation. The Mark 4 can be programmed to meet specific requirements now, and re-programmed when those requirements change. An Environ data Easi Data Mark 4 Weather Station is a robust computerized system that measures and records environmental conditions in real time. Weather station consists of four key components, external sensors that measure environmental parameters and transmit raw data, A data logger that

collects and analyses the raw data and calculates the results before storing it into internal memories, A communications method of the choice that transmits the stored data to any external computer, Environ data's software that will download and display data in preferred format. Environ data manufactures almost 20 different types of sensors to operate with the Easi Data system, These include: solar radiation, relative humidity, air temperature, wet and dry bulb temperature, grass temperature, soil temperature, rainfall, wind speed, wind direction, barometric pressure, soil moisture, radiant heat, ultra violet radiation, and Photo synthetically active radiation. Remote Communications Equipment for project include : GSM Remote Link (Global System for Mobile Communications), CDMA Remote Link (Code Division Multiple Access),Modem Dial-Up – Via a standard phone line, UHF Radio Link, Network Connection, or RS-232 Serial Cable Link – maximum preferred distance 100 meters, up to 5 km with line drivers. This project not contains system for rechargeable battery, so in isolated area not useful, and also it is high cost.

Figure 2.1.4.1 Weather Station (Easi Data Mark)

2.2 Overview of project design

This section will explain a simple design for the project by using a block diagram as shown in Figure 2.2.1 that will describe the basic idea for functionality for overall system briefly.

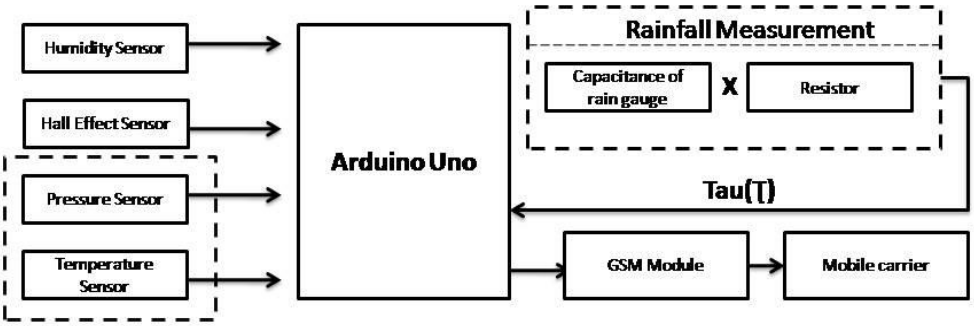


Figure 2.2.1: Block diagram for overall system

This block diagram represents all measurements which will be taken from the weather by station. The core of this design is a microcontroller (Arduino Uno) which collects all data and sends it by using GSM modem to interface circuit that will be a mobile device. The microcontroller will receive five measurements; one from these measurements is taken from rain gauge to measure the amount of rainfall, the rain gauge contains two parallel plate capacitors impacted along it, then the equivalent capacitance varies due to rainfall amount, the other measurements will be taken from four sensors; first sensor will read the temperature, second one read the humidity, and the last two sensors read speed of wind and barometric pressure.

2.3 Review of Components

This section gives a good background for all components which are used in the project and describe the function of each one.

2.3.1 Arduino Uno (ATmega328P)

The Arduino Uno is a microcontroller board, as shown in Figure 2.3.1.1 it has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a USB connection, a power jack, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

Figure 2.3.1.1 : Arduino uno board

Basic features for Arduino Uno:

- Microcontroller ATmega328
- Operating Voltage 5V
- Input voltage(recommended) 7-12 V
- Input Voltage (limits) 6-20 V
- Digital I/O Pins 14(of which 6 provide PWM output)
- Analog Input Pins 6

- DC Current per I/O Pin 40 *mA*
- DC Current for 3.3V Pin 50 *mA*
- Flash Memory 32 *KB*
- SRAM 2 *KB*
- EEPROM 1 *KB*
- Clock Speed 16 *MHz*

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Arduino Uno can be programmed using software provided, the java-based programming language is similar to C++ with some modifications, rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and over current. Although most computers provide their

own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed [8].

2.3.2 GSM Modem (SM5100B-D)

GSM Modem is a class of wireless Modem devices that are designed for communication of a computer with the GSM network. It requires a SIM (Subscriber Identity Module) card just like mobile phones to activate communication with the network. Also they have IMEI (International Mobile Equipment Identity) number similar to mobile phones, for their identification. A GSM Modem can perform the following operations:

- Receive, send or delete SMS messages in a SIM.
- Read, add, search phonebook entries of the SIM.
- Make, Receive, or reject a voice call.

The Cellular Shield for Arduino includes all the parts needed to interface your Arduino with anSM5100B cellular module as shown in Figure 2.3.2.1. This allows you to easily add SMS, GSM/GPRS, and TCP/IP functionalities to your Arduino-based project. All you need to add cellular functionality to your Arduino project is a SIM card (pre-paid or straight from your phone) and an antenna and you can start sending Serial. Print statements to make calls and send texts [9]. The main components of the Cellular Shield are a 60-pinSM5100B connector, a SIM card socket, and an SPX29302 voltage regulator configured to regulate the Arduino's Raw voltage to 3.8V. The board's red LED indicates power. The Arduino's reset button is also brought out on the shield.

Figure 2.3.2.1: GSM Modem (SM5100B-D)

AT commands are used for interface between GSM modem and Arduino board, AT commands are also known as Hayes AT commands. There are different views to understand the meanings of "AT". Some call it "Attention Telephone", whereas others call it "Attention Terminal" commands. AT commands allow giving instructions to both mobile devices and ordinary landline telephones. The commands are sent to the phone's modem, which can be a GSM modem or PC modem. Different manufacturers may have different sets of AT commands. Fortunately, many AT commands are the same. AT commands can be used for operations that are usually done from the keypad, for instance calling a number, sending, reading, or deleting an SMS, reading and deleting phonebook data, reading the battery status, reading the signal strength, and so on.

2.3.3 Humidity and Temperature Sensor – RHT03

The RHT-03 is a low cost humidity and temperature sensor with a single wire digital interface as shown in Figure 2.3.3.1. The sensor is calibrated and doesn't require extra components so you can get right to measuring relative humidity and temperature [10].

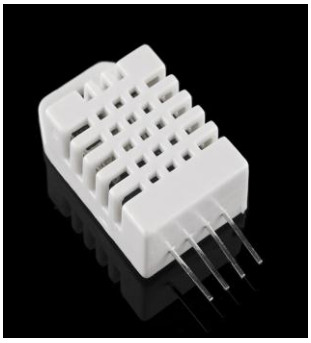


Figure 2.3.3.1: Humidity and Temperature Sensor – RHT03

Features:

- 3.3-6V Input
- 1-1.5 mA measuring current
- 40-50 μA standby current
- Humidity from 0-100% RH
- $-40-80^{\circ}C$ temperature range
- $\pm 2\%$ RH accuracy
- $\pm 0.5^{\circ}C$

2.3.4 Barometric Pressure Sensor BMP085

The BMP085 is a basic sensor that is designed specifically for measuring barometric pressure (it also does temperature measurement on the side to help). It's one of the few sensors that does this measurement, and it’s fairly low cost so you'll see it used a lot. You may be wondering why someone would want to measure atmospheric pressure, but it’s actually really useful for two things. One is to measure altitude. As we travel from below sea level to a high mountain, the air pressure decreases. That means that if we measure the pressure we can determine our altitude - handy when we don't want the expense or size of a GPS unit.

Secondly, atmospheric pressure can be used as a predictor of weather which is why weather-casters often talk about "pressure systems", BMP085 sensor is shown in Figure 2.3.4.1:

Figure 2.3.4.1: Barometric Pressure Sensor BMP085

Specifications

- Pressure sensing range: 300-1100 hPa (9000m to -500m above sea level)
- Up to 0.03hPa / 0.25m resolution
- -40 to +85°C operational range, +-2°C temperature accuracy
- 2-pin i2c interface on chip
- V1 of the breakout uses 3.3V power and logic level only
- V2 of the breakout uses 3.3-5V power and logic level for more flexible usage

2.3.5 Hall Effect Sensor

A Hall Effect sensor as shown in Figure 2.3.5.1 is a transducer that varies its output voltage in response to a magnetic field. Hall Effect sensors are used for proximity switching, positioning, speed detection, and current sensing applications.

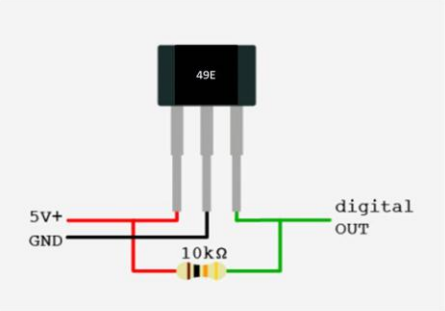


Figure 2.3.5.1: Hall Effect Sensor

2.3.6 Two Parallel Plate Capacitors

Rain gauge contains two parallel plate capacitors which are made from the stainless steel metal which was chosen, because it is a very good conductor.

It is used to calculate the amount of rain fall by the changing in the capacitance value which useful in determines this amount easy, and this shown as Figure 2.3.6.1:

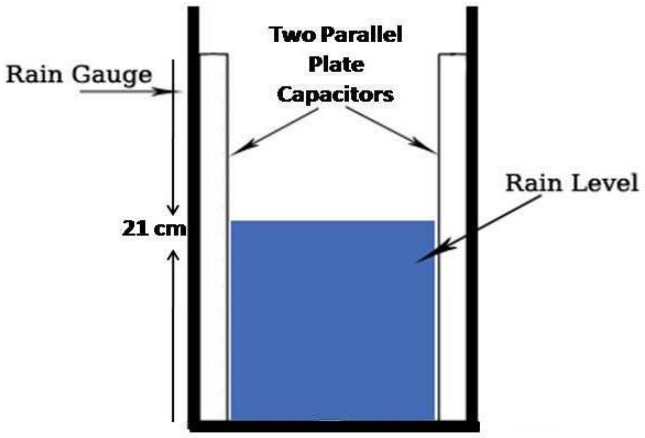


Figure 2.3.6.1: The Rain Gauge

2.3.7 Water Pump

The water pump is used to discharge the rain gauge from the water when it is full and every day. The pump operates with 12v.

2.3.8 Relay Bestar BS-115C

The relay which is used its magnetic coil operates with 5 v input to close the circuit which operates with 12v as pump water, and this relay shown as figure 2.3.8.1 below:

Figure 2.3.8.1: Relay Bestar BS-115C

2.3.9 NPN switching transistor 2N2222A

It is a switching transistor with high current (max. 800 mA), and operate at low voltage and with maximum voltage 40 V and we use it to amplify the output voltage comes from the arduino pin to supply it to the water pump. this transistor is shown in figure 2.3.9.1:

Figure 2.3g.9.1: NPN switching transistor 2N2222A

CHAPTER 3

Methodology and Design

CHAPTER 3

Methodology and Design

3.1 Mechanism of Rain Gauge Measurement

According to the amount value of the rainfall which measured by the Palestinian metrological center which are taken in the accounts to choose the best design for the size and the shape of the rain gauge , and this value was 833mm during winter semester in 2011- 2012 year. The height of rain gauge is 30 cm with diameter 4 cm as shown in Figure 3.1.1:

Figure 3.1.1: Rain Gauge Characteristics

The main idea for this measurement depends on the changing in the capacitance value for the two parallel plate capacitors which are impacted inside the rain gauge, and this value increasing linearly as rainfall level increase according to this relation:

$$C = \frac{\varepsilon_o \varepsilon_r A}{d}$$

Where:

C: The Capacitance value.

$\varepsilon_\circ$: Permittivity of free space $= 8.854 \times 10^{-12} = \frac{1}{36\pi} 10^{-9} F/M$

ε_r : Permittivity of water $= 80 F/M$.

d : The distance between two parallel plates.

A : Area for the flooded part of the parallel plate capacitors.

From the previous relation the capacitance value is inversely proportional to the distance between two parallel capacitors and this distance is a constant, so that the capacitance value depend only on the area of the capacitors which flooded in the water comes from the rainfall . The total capacitance for two parallel plate capacitors when there is no water, it is a free space capacitance and this value is very small, but when there is water inside rain gauge the total capacitance becomes as this equation:

$$C_{total} = C_1 + C_2$$

Where:

C_1: The capacitance for folded part

C_2 : The capacitance for free space part

To calculate the amount of rainfall according to the changing of the total capacitance value and using the Arduino Board which will calculate the time constant (τ) which is equal to RC_{total} ,such that R is $10\text{M}\Omega$ Resistor.

3.1.1 Rain Gauge Characteristics

To achieve the best calculations for the amount rainfall, the rain gauge will put inside the cylindrical plastic pipe to isolate it from outside conditions as shown in Figure 3.1.1.1:

Figure 3.1.1.1: The Rain Gauge inside a Cylindrical Pipe

The amount of rainfall measurements will be taken every day so that the discharging of the water is very necessary to take a new reading in the next day, so the pump water is used to do this issue which placed inside the pipe.

3.1.2 Calculations and relations

To find a relation between capacitance and amount of rainfall, some calibrations are needed to achieve it, and these calibrations are divided into 8 levels with 30 ml in each one as shown in Table 3.1.2.1:

CapSens(R*C)	Water level(ml)
23547	0
24316	30
26382	60
28133	90
30293	120
32631	150
34673	180
37335	210

Table 3.1.2.1: Relation between CapSens and Amount of Rainfall

From previous values the relation as shown in Figure 3.1.2.2:

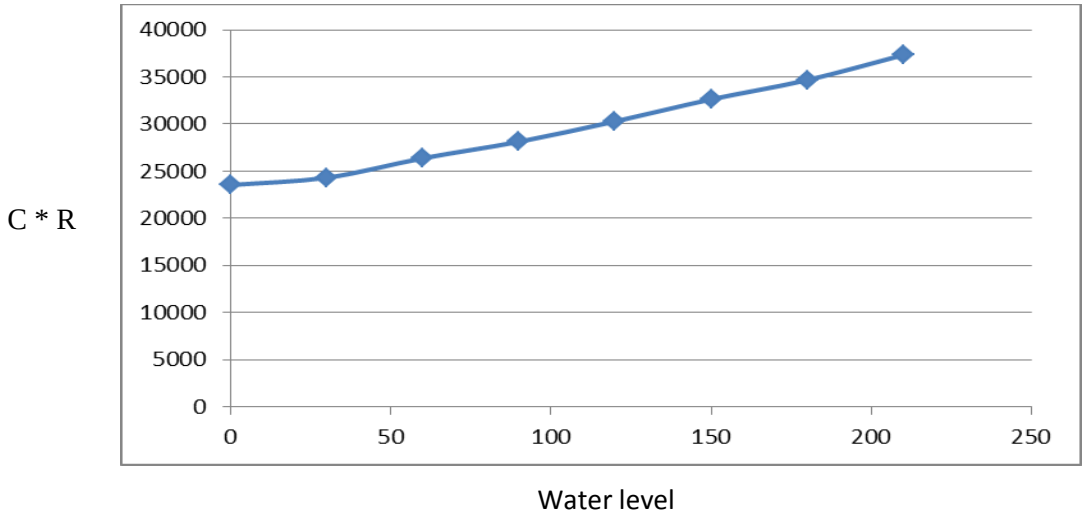


Figure 3.1.2.2: The relation between the water level And the (capacitance x resistor)

The relation between water level and capacitance approximately linearly, so that the relation according for this as follows:

$$X = \frac{Y - 21593.6}{72.66}$$

Where:

X: Amount of rainfall.

Y: capacitance x resistor.

To calculate the exact value for how much amount rainfall using the rain gauge, we have been made the celebration in a way that takes into account the volume of the two parallel plates and the hose, to do this we first put the water into the gauge and after that we put the two parallel plates and the hose together into it, then we take the reading of the gauge and write our notes.

3.1.3 Interface with Arduino Board

The interface between the two parallel plate capacitors and the arduino board is shown in Figure 3.1.3.1:

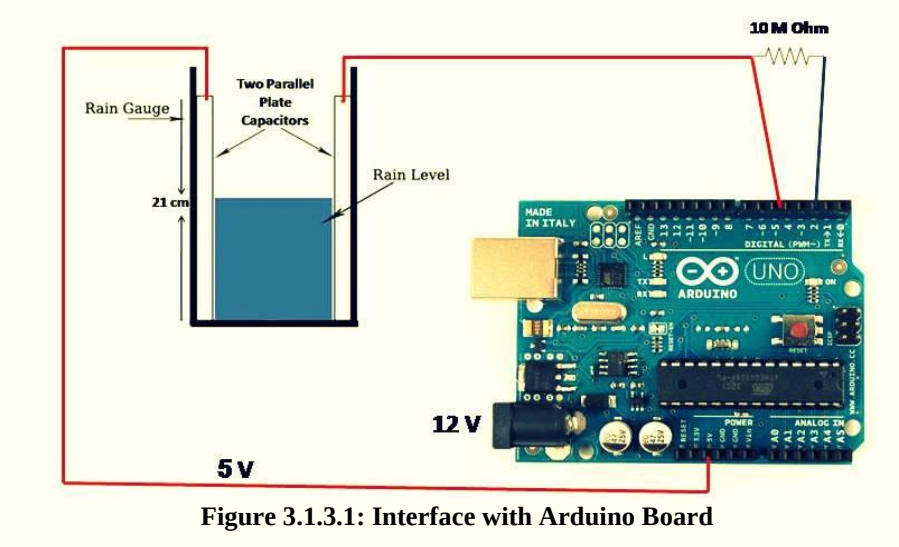


Figure 3.1.3.1: Interface with Arduino Board

3.2 Sensors Interface and Connections

This section shows the interfaces and connections for all sensors with Arduino Uno.

3.2.1 Humidity and Temperature Sensor – RHT03

The RHT03 sensor works from 3.3v to 6v, pin1 is connected to 5v on Arduino. The sensor pin 2 goes to the Arduino pin 5, with a 1Kohm pull-up resistor as shown in Figure 3.2.1.1:

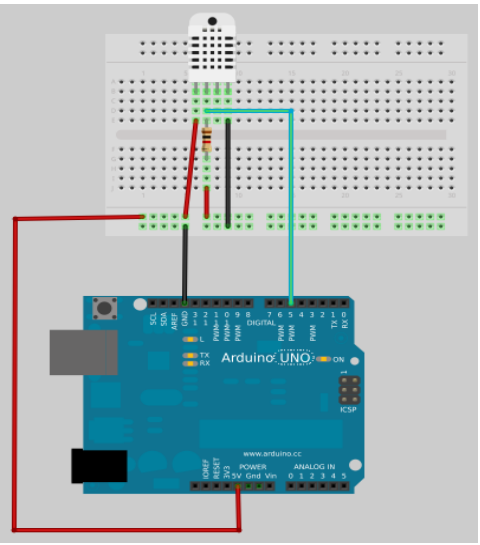


Figure 3.2.1.1: Humidity and Temperature Sensor Interface

3.2.2 Barometric Pressure Sensor BMP085

Barometric Pressure Sensor BMP085 connection is shown in Figure 3.2.2.1,

- We Connect the VCC pin to a 3.3V power source. The V1 of the sensor breakout cannot be used with anything higher than 3.3V so we don't use a 5V supply, V2 of the sensor board has a 3.3V regulator so we can connect it to either 3.3V or 5V if we do not have 3V available.
- We Connect GND to the ground pin.

- We connect the i2c SCL clo ck pin to our i2c clock pin(Analog pin #5).
- We Connect the i2c SDA data pin to your i2c data pin(Analo g pin #4)
- Unfortunately, the i2c lines on most microcontrollers are fixed so we are going to have to stick with those pins.
- We don't need to connect the XCLR (reset) or EOC (end-of-conversion) pins. If we need to speed up our conversion time, we can use the EOC as an indicator - in our code we just hang out and wait the maximum time possible.

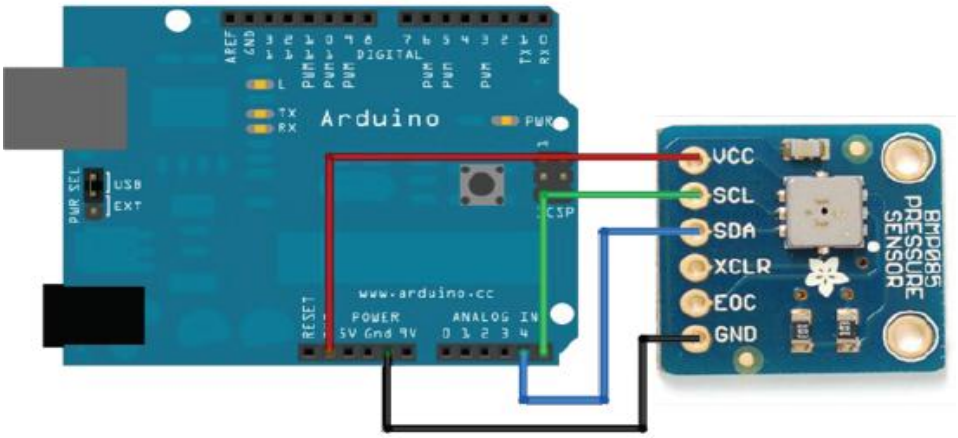


Figure 3.2.2.1: Barometric Pressure Sensor BMP085

3.2.3 Hall Effect Sensor

Hall effect sensor has three pins, as shown in figure pin1 is connected to digital output, pin2 to ground , pin3 to 5v and $10K\Omega$ resistor between pin1 and pin3 as shown in Figure 3.2.3.1:

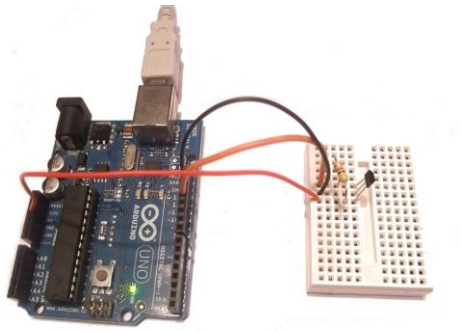


Figure 3.2.3.1: Hall Effect Sensor.

3.3 Mechanism of Wind Speed Measurements

In this section the mechanism of wind speed measurements will be discussed and these measurements depend with its operation on the magnetic field, so that the Hall Effect sensor to sense the magnetic field and the magnet will be used to do this operation**.**

To calculate the wind speed the end plastic PVC cab is used as follows in Figure 3.3.1:

Figure 3.3.1: Plastic PVC End Cab

The Hall Effect sensor will locate on the main column of the anthropomorphic, and the magnet will be put in versus with the Hall Effect sensor by putting it in one of the four arms of the PVC end as shown in figure 3.3.2:

Figure 3.3.2: Hall Effect Sensor and magnet Location

The main idea to calculate the number of the revolutions per unit time so that to calculate it, every one revolution the magnet cross the sensor and the sensor will be high then is very is to calculate how many revolution per unit time by calculating how many times the sensor be high every one minute to calculate the angular velocity ω which is equal to:

$$\omega = \frac{n}{t}$$

Where:

n: Numbers of revolutions (rad).

t: Time (t).

Now to calculate the wind speed, this angular velocity will be transferred to be as a linear velocity v as this formula:

$$v = \omega.R$$

Where:

ω: Angular Velocity (rad/s).

R: Radius from the center of the end cap to the center of the end cups (m).

3.4 GSM Modem Interface with Arduino Board

The interface between GSM modem and Arduino Uno is very easy, as shown in Figure 3.4.1 all pins for Arduino the same pins on GSM modem. The GSM modem can often require up to 2A of current and it needs 5V power supply but the Arduino board can only supply up to just under 1A, so that it is highly recommended to use an external 5V power supply capable of delivering 2A of current from an AC adapter

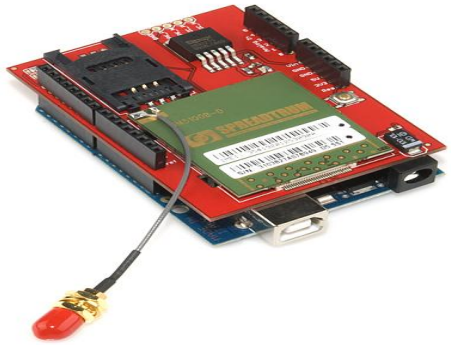


Figure 3.4.1: Interface between GSM Modem and Arduino

CHAPTER 4

Results

4.1 About Sensors Measurements.

4.2 About Rain Gauge Measurements.

4.3 About Arduino Uno and GSM Modem.

4.1 About Sensors Measurements

According to datasheet got for sensors, the properties for these sensors are shown in Table 4.1.1:

Sensors Names	Operating Ranges	Output	ERROR
Humidity And Temperatures Sensor	0-100%	Digital	+-2% RH +-0.5 C
Pressure	0-1023	Analog	1.5%
Hall effect	0-1023	Digital	5%

Table 4.1.1: Sensor specifications

The different output readings which was taken from the sensors was the same ranges in the previous table and don't exceed it, and the Figure 4.1.2 show some these output readings from all sensors after was connected.

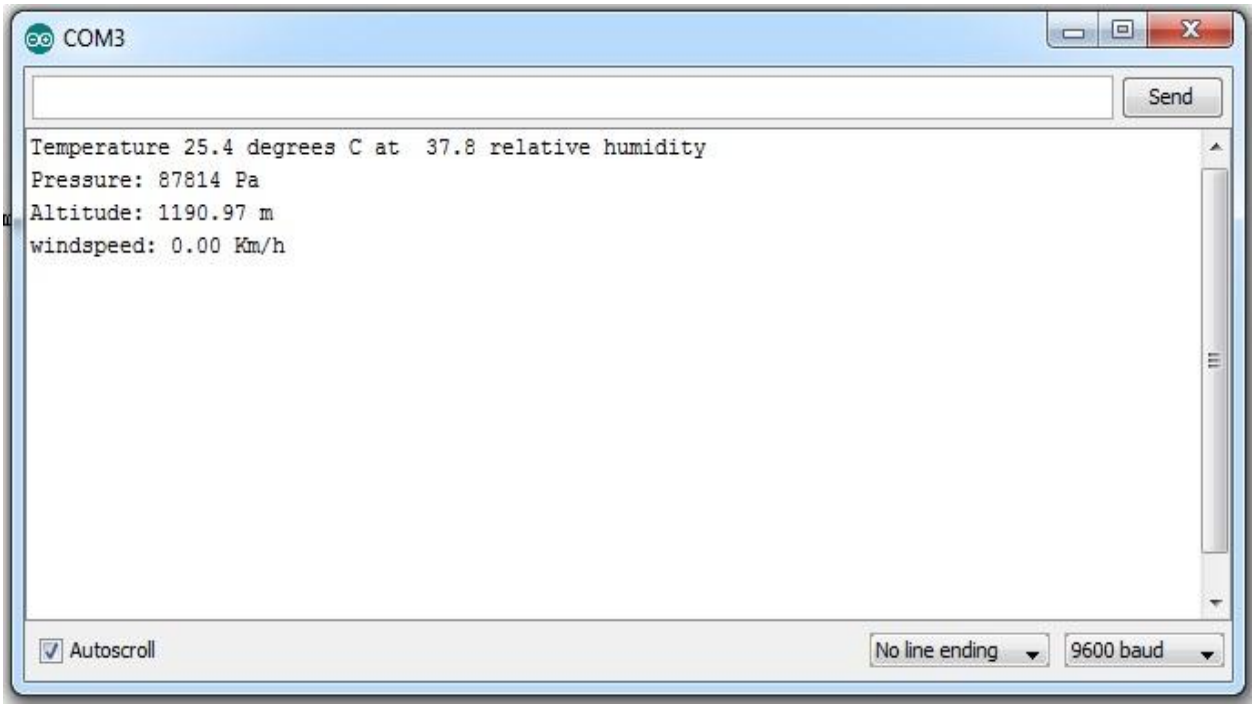


Figure 4.1.2: Output Readings of Sensors

4.2 About Rain Gauge Measurements

In the following figure we show some trials and experiments to find the relation between the water level and the value of R*C Figure 4.2.1:

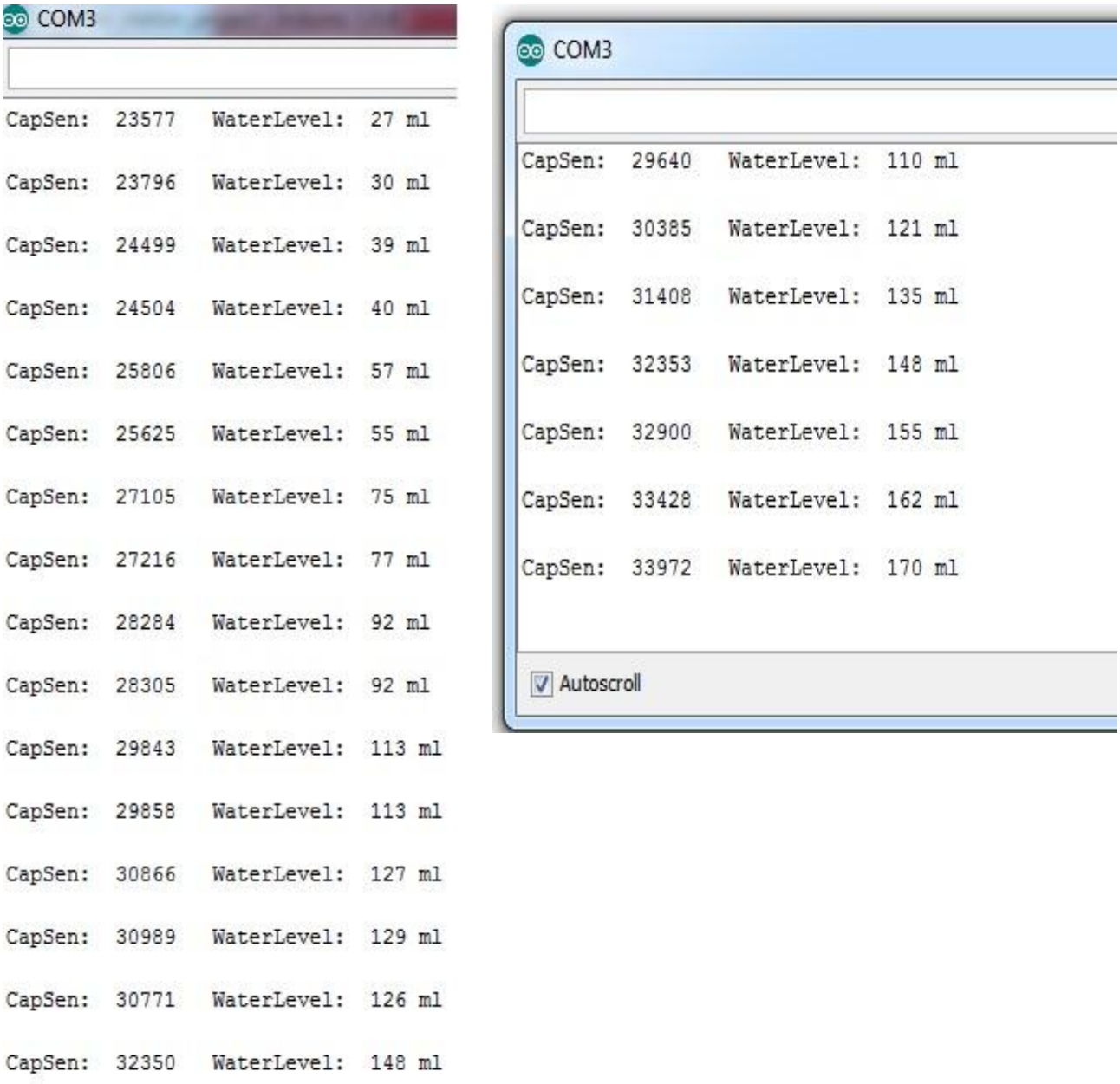


Figure 4.2.1: Output of water level and the CapSens (R*C)

By making a step by step of water adding into the gauge with 30 ml for each step we find that the relation between the water level and the value of CapSens(R*C) is given by the following figure 4.2.2

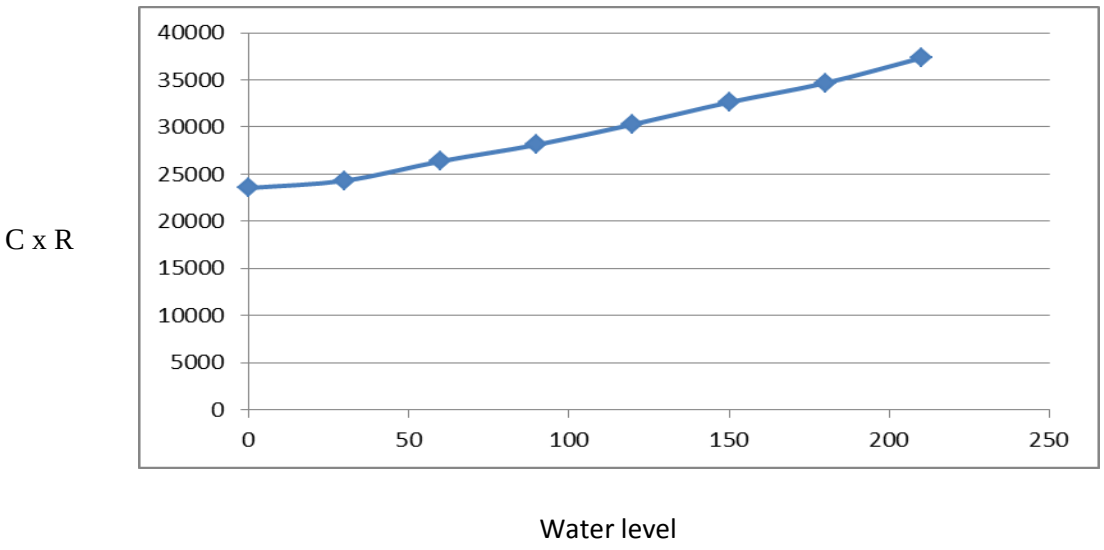


Figure 4.2.2: The relation between water level and the CapSens

4.3 About Arduino Uno and GSM Modem

The Arduino uno was very easy to handle and designed in a way that allows it to be reset by software running on a connected computer. In addition it can interface with many devices and components such as GSM Modem.

CHAPTER 5

Discussion and Conclusion

5.1 Problems we faced

5.2 Future improvements

5.3 Conclusion

5.4 Commercial Study

5.1 Problems We Faced

We faced a lot of problems during the work but we will discuss main it:

- Some components not available in the market, so that it was changed with other components.
- What is the best design for our base station?
- How will the design wind speed sensor without friction to move more easily?
- Some Components are expensive.
- The two parallel capacitors are very sensitive for the outside conditions.

5.2 Future improvement

- Install a weather station with a renewable energy.
- Make a network for weather station in different areas.
- Display the data through website weather.
- Use more sensitive components to get more reliable data.

5.3 Conclusion

Design wireless base station to collect the data from the weather and send them through GSM Modem to Mobile Carriers without needing for Meteorological observer.

5.4 Commercial Study

In our project we tried to use components which have more needed functionality with high efficiency and minimum cost. The Table 5.4.1 below shows the cost of each component used in the project.

Name of component	**Cost(NIS)**
Arduino Uno	205
GSM Modem SM5100B-D	570
Humidity and Temperature Sensor RHT03	73
Barometric Pressure Sensor BMP085	80
Rain Gauge	30
Hall Effect Sensor	25
Water Pump	25
Relay	7
Transistor	1
Bottle	12

Table 5.4.1: Cost of Components

Appendix

Appendix A: ARDUINO UNO ATMEGA328P

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)
  - 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
  - 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator

8-bit AVR® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash

  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 20 MHz @ 1.8 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.75 µA (Including 32 kHz RTC)

Appendix B: GSM MODEM

SPREADTRUM

SM5100B-D GSM/GPRS Module Specification (Preliminary)

Version: 1.0.0

HW-SM5100B-D-DS-0002

2006-4-4

Connection	60 pins
Power supply	VBAT: 3.3V to 4.2V range, 3.6V typical.
Power consumption	Off mode: <100uA Sleep mode: <2.0mA Idle mode: <7.0mA (average) Communication mode: 350 mA (average,GSM) Communication mode: 2000mA (Typical peak during TX slot,GSM)
Li-ion Battery charging management and interface (OPTION)	Li-ion Battery charging management is included. The charger interface is provided on 60-pin connector. (only for 3.7V Li-ion Battery)
Frequency bands	EGSM900 +GSM850+ DCS1800+PCS1900
Transmit power	Class 4 (2W) for EGSM900/GSM850 Class 1 (1W) for DCS1800/PCS1900
Supported SIM card	3V/1.8V SIM card. (auto recognise)
Keyboard interface	4x6 keyboard interface is provided
UART0 interface with flow control	Up to 460 kbps Full hardware flow control signals (+3.0V) are provided on 60 pins.
UART1 interface without flow control	2-Wire UART interface Up to 460 kbps

2.4.2 SIM Interface

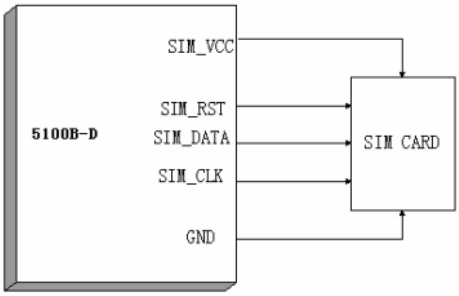
Table 11: SIM Pin Description

Pin number	Pin name	Type	Description
29	SIM_DA	I/O	SIM Serial Data
27	SIM_CLK	O	SIM Clock
21	SIM_RST	O	SIM Reset
51	SIM_VCC	P	SIM Power Supply

Table 12: SIM Electrical Characteristics

parameter	Min	Max	Remarks
VL (V)	-	0.6/0.8	
VH (V)	1.3/2.2	-	

SIM card connection:



4.1.3 Antenna

Antenna structure and integration in the application is a major issue.

Attention should be paid to:

- Selection of the antenna cable (performance, length, type, thermal resistance, etc.)
- Selection of antenna connector (type, losses, mismatch, etc.)
- Antenna isolation from digital circuits (including the interface signals)

A poor antenna could dramatically affect the GSM performance such as sensitivity and transmit power.

Warning:
Spreadtrum strongly recommends working with an antenna manufacturer either to develop an antenna for the application or to adapt an existing solution to the application. The antenna adaptation (mechanical and electrical adaptation) is one of the key issues in the design of a GSM terminal.

Attention should be paid to:

- Antenna cable integration (bend, length, position, etc.
- Legs of the module are mechanically mounted to the ground plane.

Appendix C: Humidity and Temperature Sensor – RHT03

Maxdetect
sense the world

Your specialist in innovating humidity & temperature sensors

Digital relative humidity & temperature sensor RHT03

1. Feature & Application:

*High precision
*Capacitive type
*Full range temperature compensated
*Relative humidity and temperature measurement
*Calibrated digital signal
*Outstanding long-term stability
*Extra components not needed
*Long transmission distance, up to 100 meters
*Low power consumption
*4 pins packaged and fully interchangeable

2. Description:

RHT03 output calibrated digital signal. It applys exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability. Its sensing elements is connected with 8-bit single-chip computer.

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

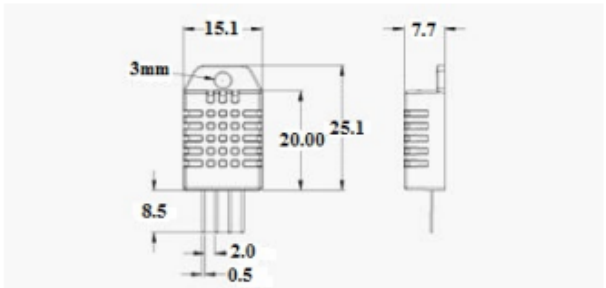
Small size & low consumption & long transmission distance(100m) enable RHT03 to be suited in all kinds of harsh application occasions. Single-row packaged with four pins, making the connection very convenient.

3. Technical Specification:

Model	RHT03	
Power supply	3.3-6V DC	
Output signal	digital signal via MaxDetect 1-wire bus	
Sensing element	Polymer humidity capacitor	
Operating range	humidity 0-100%RH;	temperature -40~80Celsius
Accuracy	**humidity +-2%RH**(Max +-5%RH);	temperature +-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH;	temperature 0.1Celsius
Repeatability	humidity +-1%RH;	temperature +-0.2Celsius

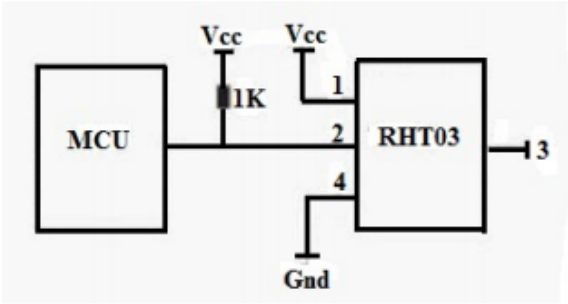
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Interchangeability	fully interchangeable

4. Dimensions: (unit----mm)



Pin sequence number: 1 2 3 4 (from left to right direction).

Pin	Function
1	VDD—power supply
2	DATA–signal
3	NULL
4	GND



5. Electrical connection diagram:

6. Operating specifications:

(1) Power and Pins

Power's voltage should be 3.3-6V DC. When power is supplied to sensor, don't send any instruction to the sensor within one second to pass unstable status. One capacitor valued 100nF can be added between VDD and GND for wave filtering.

(2) Communication and signal

MaxDetect 1-wire bus is used for communication between MCU and RHT03. (MaxDetect 1-wire bus is specially designed by MaxDetect Technology Co., Ltd. , it's different from Maxim/Dallas 1-wire bus, so it's incompatible with Dallas 1-wire bus.)

Illustration of MaxDetect 1-wire bus:

Data is comprised of integral and decimal part, the following is the formula for data.

DATA=8 bit integral RH data+8 bit decimal RH data+8 bit integral T data+8 bit decimal T data+8 bit check-sum

If the data transmission is right, check-sum should be:

Check sum=8 bit integral RH data+8 bit decimal RH data+8 bit integral T data+8 bit decimal T data

Example: MCU has received 40 bits data from RHT03 as

0000 0010 1000 1100	0000 0001 0101 1111	1110 1110
16 bits RH data	16 bits T data	check sum

Check sum=0000 0010+1000 1100+0000 0001+0101 1111=1110 1110

RH= (0000 0010 1000 1100)/10=65.2%RH

T=(0000 0001 0101 1111)/10=35.1□

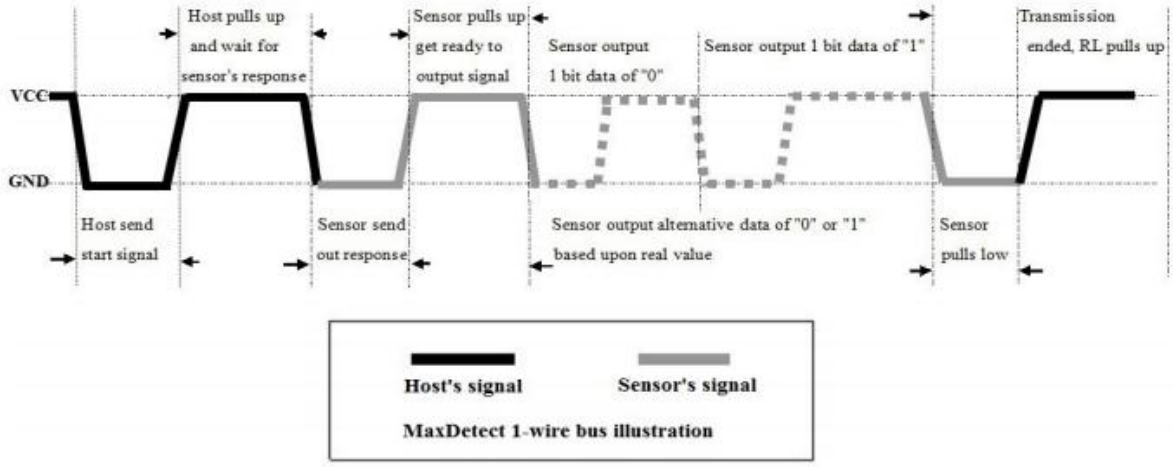
When highest bit of temperature is 1, it means the temperature is below 0 degree Celsius.

Example: 1000 0000 0110 0101, T= minus 10.1□

16 bits T data

When MCU send start signal, RHT03 change from standby-status to running-status. When MCU finishs sending the start signal, RHT03 will send response signal of 40-bit data that reflect the relative humidity and temperature to MCU. Without start signal from MCU, RHT03 will not give response signal to MCU. One start signal for one response data from RHT03 that reflect the relative humidity and temperature. RHT03 will change to standby status when data collecting finished if it don't receive start signal from MCU again.

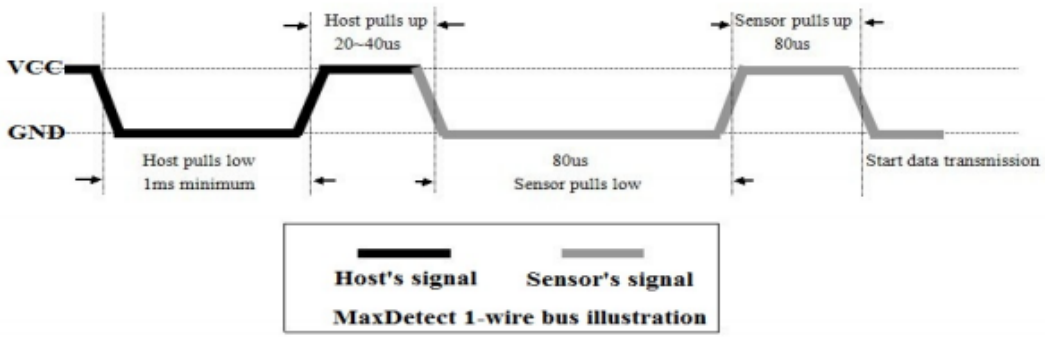
See below figure for overall communication process, **the interval of whole process must beyond 2 seconds.**



1) Step 1: MCU send out start signal to RHT03 and RHT03 send response signal to MCU

Data-bus's free status is high voltage level. When communication between MCU and RHT03 begins, MCU will pull low data-bus and this process must beyond at least 1~10ms to ensure RHT03 could detect MCU's signal, then MCU will pulls up and wait 20-40us for RHT03's response.

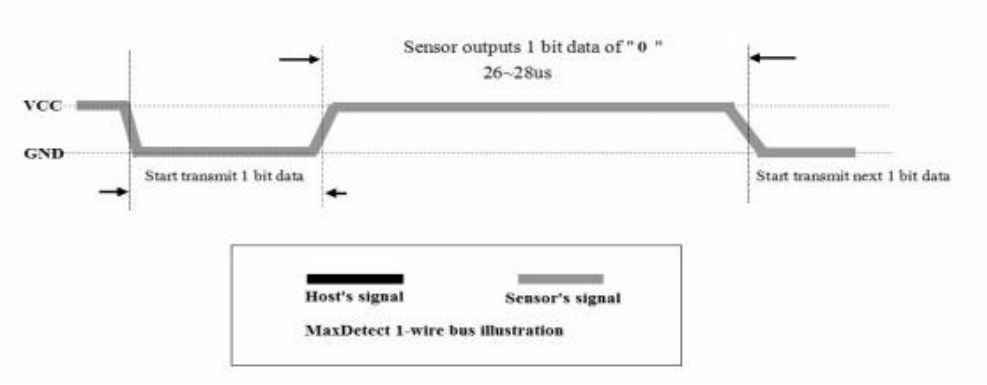
When RHT03 detect the start signal, RHT03 will pull low the bus 80us as response signal, then RHT03 pulls up 80us for preparation to send data. See below figure:

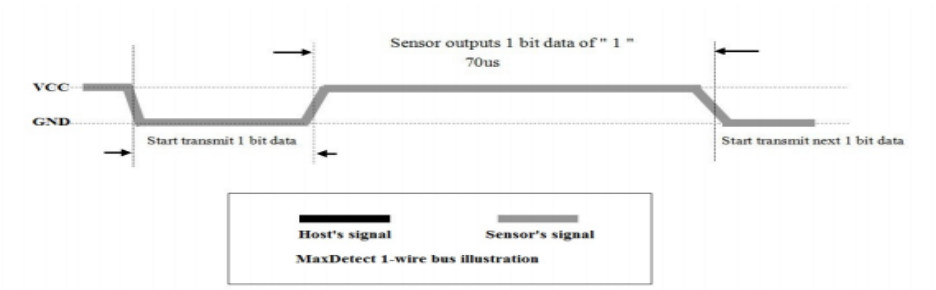


---

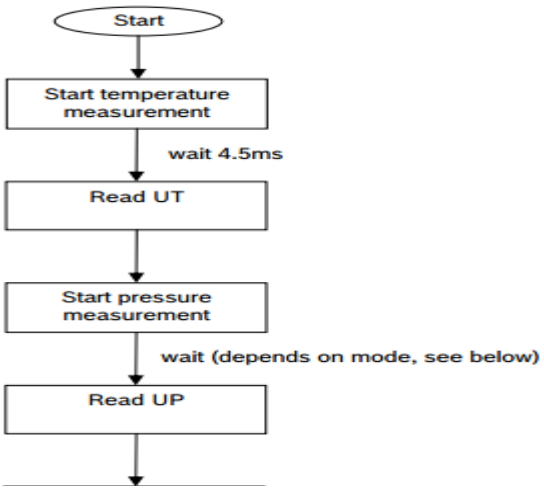
2). Step 2: RHT03 send data to MCU

When RHT03 is sending data to MCU, every bit's transmission begin with low-voltage-level that last 50us, the following high-voltage-level signal's length decide the bit is "1" or "0". See below figures:





Appendix D: Barometric Pressure Sensor BMP085

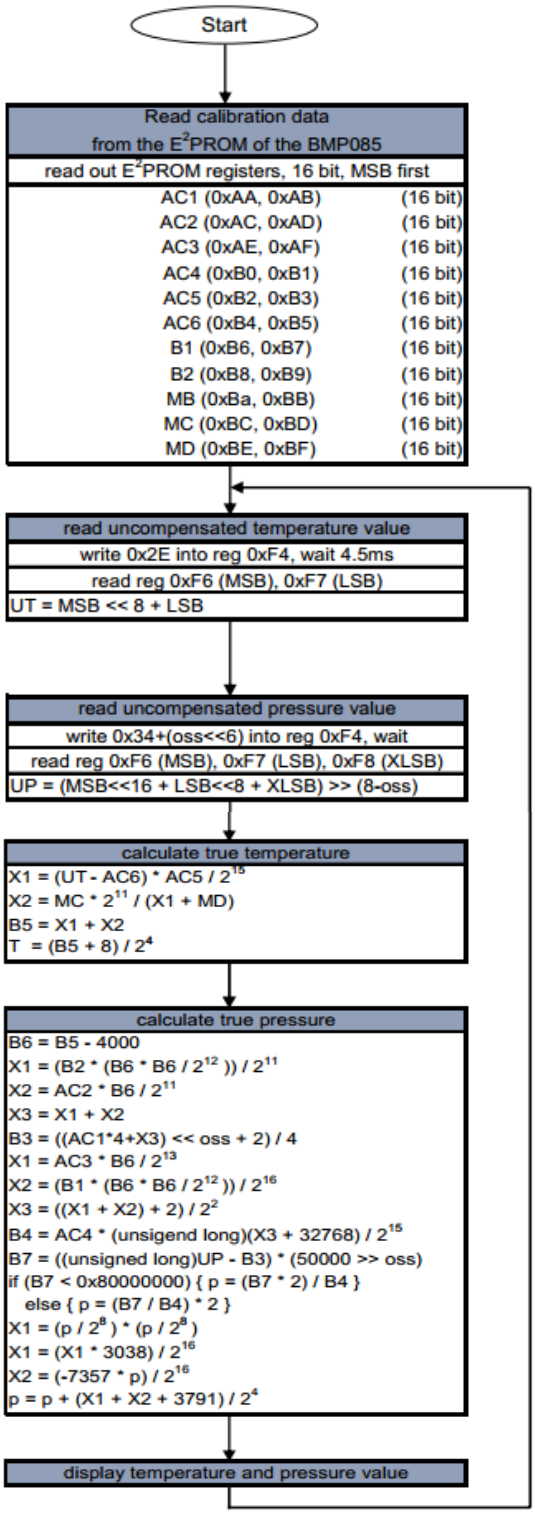


Calibration coefficients

The 176 bit E^2PROM is partitioned in 11 words of 16 bit each. These contain 11 calibration coefficients. Every sensor module has individual coefficients. Before the first calculation of temperature and pressure, the master reads out the E^2PROM data.
The data communication can be checked by checking that none of the words has the value 0 or 0xFFFF.

	BMP085 reg adr	
Parameter	MSB	LSB
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF

Calculation of pressure and temperature for BMP085



	example:	C code function:	type:
		bmp085_get_cal_param	
AC1 =	408		short
AC2 =	-72		short
AC3 =	-14383		short
AC4 =	32741		unsigned short
AC5 =	32757		unsigned short
AC6 =	23153		unsigned short
B1 =	6190		short
B2 =	4		short
MB =	-32767		short
MC =	-8711		short
MD =	2868		short
		bmp085_get_ut	
UT =	**27898**		long
oss =	0		
= *oversampling_setting*			short (0 .. 3)
(ultra low power mode)		bmp085_get_up	
UP =	**23843**		long
		bmp085_get_temperature	
X1 =	4743		long
X2 =	-2344		long
B5 =	2399		long
T =	**150**	***temp in 0.1°C***	long
		BMP085_calpressure	
B6 =	-1601		long
X1 =	1		long
X2 =	56		long
X3 =	57		long
B3 =	422		long
X1 =	2810		long
X2 =	59		long
X3 =	717		long
B4 =	33457		unsigned long
B7 =	1171050000		long
p =	70003		long
			long
X1 =	74774		long
X1 =	3466		long
X2 =	-7859		long
p =	**69965**	***press. in Pa***	long

Appendix E: Relay Bestar BS-115C

BS-115C *MINIATURE RELAY* *Bestar*

TÜV

■ FEATURES

- Ultra-miniature size (18.8 *15.4 *15.1 mm)
- High Switching Capacity up to 12A
- UL File No. E147052
- CSA File No. LR76479-1
- TUV File No. R9754066

■ COIL RATING (at 20°C)

Nominal Voltage (VDC)	Coil Resistance (Ω±10%)	Nominal Current (mA)	Pick-Up Voltage (VDC)	Drop-Out Voltage (VDC)	Nominal Power (mW) Consumption
3	25	120	2.25	0.3	360
5	70	72	3.75	0.5	
6	100	60	4.50	0.6	
9	220	40	6.75	0.9	
12	400	30	9.00	1.2	
24	1600	15	18.00	2.4	
48	6400	7.5	36.00	4.8	

■ ORDERING INFORMATION

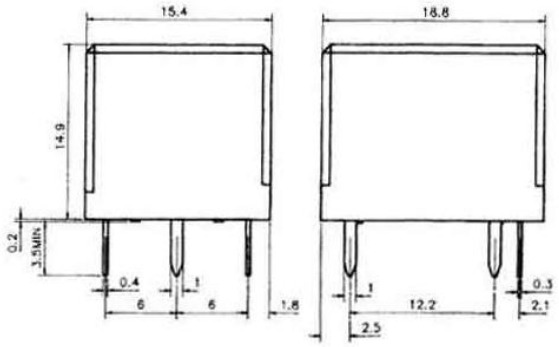
• BS-115C S-A-12A-12VDC

Protection	Contact Arrangement	Contact Rating	Coil Voltage
Nil: Flux Free S : Sealed	Nil: 1 Form C A: 1 Form A	7A 12A	See Coil Rating

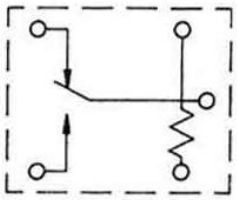
■ SPECIFICATIONS

Model No.	BS-115C-7A	BS-115C-12A
Contact Arrangement	1 Form A, 1 Form C	
Contact Material	AgCdO	
Contact Rating (at Resistive Load)	7A 120VAC 5A 240VAC/28VDC	12A 120VAC 10A 240VAC/28VDC
Max. Switching Voltage	240VAC, 60VDC	
Max. Switching Current	7A	12A
Min. Switching Load	100mA 5VDC	
Contact Resistance	Max. 100mΩ (initial)	
Insulation Resistance	Min. 100MΩ at 500VDC	
Dielectric Strength Between Contacts Between Coil & Contact	 750VAC 50 HZ/60 HZ (1 minute) 1500VAC 50 HZ/60 HZ (1 minute)	
Surge Strength	2000V	
Operate Time	Max. 10mSec.	
Release Time	Max. 5mSec.	
Ambient Temperature	-30°C ~ +55°C	
Vibration Resistance (Endurance)	1.5mm D.A. 10-55HZ	
Shock Resistance	Min. 10G Unerror	
Mechanical Life	5,000,000 Operations (no load)	
Electrical Life	100,000 Operations (at rated load)	
Weight	Approx. 8g	

DIMENSIONS (in mm)

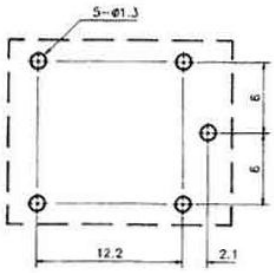


WIRING DIAGRAM
(Bottom View)



DRILLING PLAN (in mm)
(Bottom View)

General Tolerance +-0.3



Appendix F: NPN switching transistor 2N2222A

Watertown, MA
Microsemi
Progress Powered by Technology

580 Pleasant St.
Watertown, MA 02172
PH: (617) 926-0404
FAX: (617) 924-1235

2N2222A

75 Volts
0.8 Amps

NPN
BIPOLAR
TRANSISTOR

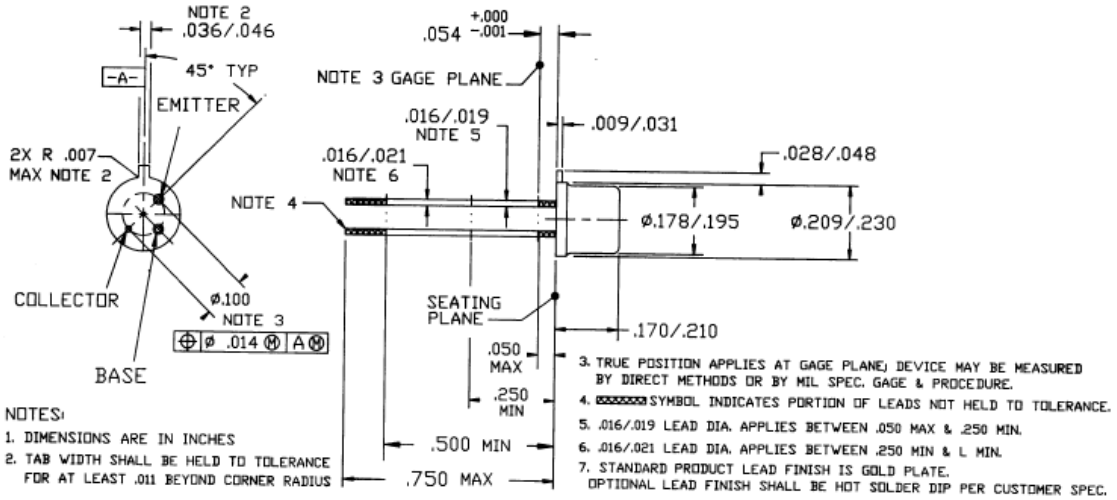
Features

- Meets MIL-S-19500/255
- Collector-Base Voltage 75
- Collector Current: 800mA
- Fast Switching 335 nS

Maximum Ratings

RATING	SYMBOL	MAX.	UNIT
Collector-Emitter Voltage	V_{CEO}		Vdc
Collector-Base Voltage	V_{CBO}	75	Vdc
Emitter-Base Voltage	V_{EBO}	6.0	Vdc
Collector Current	I_C	800	mAdc
Total Device Dissipation @ T_A = 25°C Derate above 25°C	P_D	0.5 2.85	Watt mW/°C
Total Device Dissipation @T_C = 25°C Derate above 25°C	P_D	1.8 10.3	Watt mW/°C
Thermal Resistance, Junction to Ambient	$R_{\theta JA}$	350	°C/W
Thermal Resistance, Junction to Case	$R_{\theta JC}$	97	°C/W
Operating Temperature Range	T_J	-65 to + 200	°C
Storage Temperature Range	T_{STG}	-65 to + 200	°C

Mechanical Outline



Electrical Parameters (T_A @ 25°C unless otherwise specified)

CHARACTERISTICS	SYMBOL	MIN.	TYP.	MAX.	UNIT
Off Characteristics					
Collector-Emitter Breakdown Voltage (I_C = 10 mAdc, I_B = 0)	BV_{CE0}	50		--	Vdc
Collector-Emitter Breakdown Voltage (I_C = 10 µAdc, I_E = 0)	BV_{CBO}	75		--	Vdc
Emitter-Base Breakdown Voltage (I_E = 10 µAdc, I_C = 0)	BV_{EBO}	6.0		--	Vdc
Collector to emitter Cutoff Current (V_{CE} = 30 Vdc)	I_{CES}	--		50	nAdc
Collector to base Cutoff Current (V_{CE} = 60 Vdc)	I	--		10	nAdc
D.C. Current Gain	h_{FE}				
(I_C = 0.1 mAdc, V_{CE} = 10 Vdc)		50		--	
(I_C = 1.0 mAdc, V_{CE} = 10 Vdc)		75		325	
(I_C = 10 mAdc, V_{CE} = 10 Vdc)(1)		100		--	
(I_C = 10 mAdc, V_{CE} = 10 Vdc, T_A = -55°C)(1)		35		--	
(I_C = 150 mAdc, V_{CE} = 10 Vdc)(1)		100		300	
(I_C = 500 mAdc, V_{CE} = 10 Vdc)(1)		30		--	
Collector-Emitter Saturation Voltage(1)	$V_{CE(Sat)}$				Vdc
(I_C = 150 mAdc, I_B = 15 mAdc)		--		0.3	
(I_C = 500 mAdc, I_B = 50 mAdc)		--		1.0	
Base-Emitter Saturation Voltage(1)	$V_{BE(Sat)}$				Vdc
(I_C = 150 mAdc, I_B = 15 mAdc)		0.6		1.2	
(I_C = 500 mAdc, I_B = 50 mAdc)		--		2.0	
Current Gain-Bandwidth Product(2)	f_T				Mhz
(I_C = 20 mAdc, V_{CE} = 20 Vdc, f = 100MHz)		250		--	
Output Capacitance(3)	C_{OBO}				pf
(V_{CB} = 10 Vdc, I_E = 0, 100kHz ≤ f ≤ 1MHz		--		8.0	
Input Capacitance	C_{IBO}				pf
(V_{EB} = 0.5 Vdc, I_C = 0, 100kHz ≤ f ≤ 1MHz)		--		25	
Switching Characteristics	t_{ON}				ns
Delay Time: (V_{CC} = 30 Vdc, $V_{BE(off)}$ = -0.5 Vdc,	t_d	--		10	
Rise Time: I_C = 150 mAdc, I_{B1} = 15 mAdc)(Figure 12)	t_r	--		25	
	t_{off}				
Storage Time: (V_{CC} = 30 Vdc, I_C = 150 mAdc,	t_s			225	
Fall Time: I_{B1} = I_{B2} = 15 mAdc)	t_f	--		60	

Appendix G: Hall Effect Sensor

4 Absolute Maximum Ratings

Parameter	Symbol	Value	Units
Supply Voltage	V_{DD}	28	V
Supply Current	I_{DD}	50	mA
Output Voltage	V_{OUT}	28	V
Output Current	I_{OUT}	50	mA
Storage Temperature Range	T_S	-50 to 150	°C
Maximum Junction Temperature	T_J	165	°C

Table 1: Absolute maximum ratings

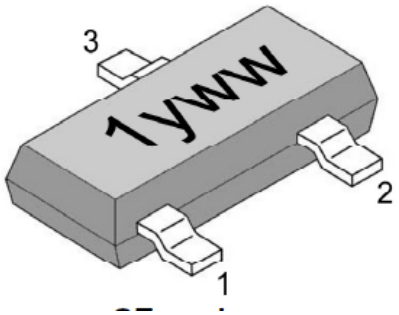
Exceeding the absolute maximum ratings may cause permanent damage. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

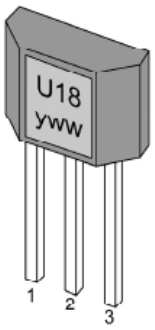
Operating Temperature Range	Symbol	Value	Units
Temperature Suffix "E"	T_A	-40 to 85	°C
Temperature Suffix "K"	T_A	-40 to 125	°C
Temperature Suffix "L"	T_A	-40 to 150	°C

5 Pin Definitions and Descriptions

SE Pin №	UA Pin №	Name	Type	Function
1	1	VDD	Supply	Supply Voltage pin
2	3	OUT	Output	Open Drain Output pin
3	2	GND	Ground	Ground pin

Table 2: Pin definitions and descriptions



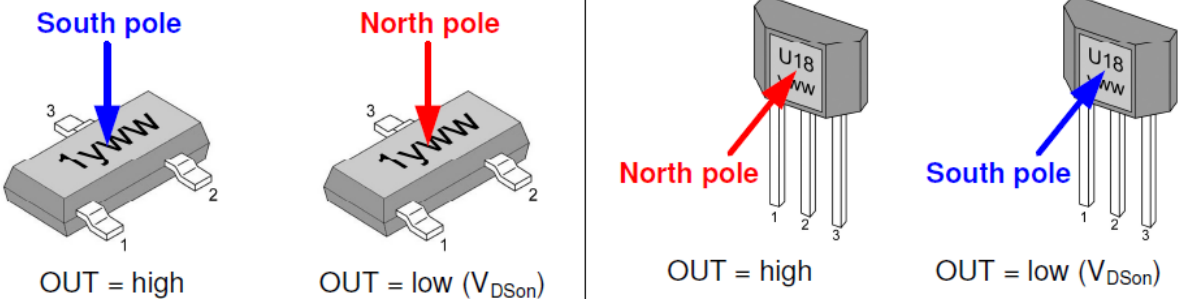


8 Output Behaviour versus Magnetic Pole

DC Operating Parameters T_A = -40°C to 150°C, V_{DD} = 3.5V to 24V (unless otherwise specified)

Parameter	Test Conditions (SE)	OUT (SE)	Test Conditions (UA)	OUT (UA)
South pole	B < B_{RP}	High	B > B_{OP}	Low
North pole	B > B_{OP}	Low	B < B_{RP}	High

Table 5: Output behaviour versus magnetic pole



9 Detailed General Description

Based on mixed signal CMOS technology, Melexis US1881 is a Hall-effect device with high magnetic sensitivity. This multi-purpose latch suits most of the application requirements.

The chopper-stabilized amplifier uses switched capacitor technique to suppress the offset generally observed with Hall sensors and amplifiers. The CMOS technology makes this advanced technique possible and contributes to smaller chip size and lower current consumption than bipolar technology. The small chip size is also an important factor to minimize the effect of physical stress.
This combination results in more stable magnetic characteristics and enables faster and more precise design.

The wide operating voltage from 3.5V to 24V, low current consumption and large choice of operating temperature range according to “L”, “K” and “E” specification make this device suitable for automotive, industrial and consumer applications.

The output signal is open-drain type. Such output allows simple connectivity with TTL or CMOS logic by using a pull-up resistor tied between a pull-up voltage and the device output.

Appendix H: System Code

```
////////////////////////////////////////////////// including libraries needed

#include <Wire.h>

#include <CapacitiveSensor.h>

#include <stdio.h>

#include <util/delay_basic.h>

#include <Time.h>

#include <TimeAlarms.h>

#include <SoftwareSerial.h>

#define BMP085_ADDRESS 0x77

SoftwareSerial cell(10,11);

// //////////////////////////////////////////////definetion for some variables

char phoneNumber[] = "00972568607003";

float sensors[5];

char globalbuf[128];

int WaterLevel=0;

int Rev=0;;

float windspeed=0;

CapacitiveSensor cs_4_6 = CapacitiveSensor(4,6);

const unsigned char OSS = 0;

int ac1;

int ac2;

int ac3;

unsigned int ac4;

unsigned int ac5;

unsigned int ac6;

int b1;

int b2;

int mb;
```

```
int mc;
int md;

long b5; // b5 is calculated in bmp085GetTemperature(...), this variable is also used in bmp085GetPressure(...)
      // so ...Temperature(...) must be called before ...Pressure(...).
short temperature;
long pressure;
const float p0 = 101325;
float altitude;
const int sensorReadTimeoutMillis = 1000;
const int serialBaudRate = 9600;
const int sensorReadIntervalMs = 5000;
int millisSinceLastRead = 0;
int sensorPin = 2;
int pum=9;
volatile long lastTransitionMicros = 0;
volatile int signalLineChanges;
int timings[88];
char errorMsgBuf[256];
boolean errorFlag = false;
////////////////////////////////////////////////////////////////////////////////////////////////
void bmp085Calibration()//// calibration function for BMP085 sensor
{
  ac1 = bmp085ReadInt(0xAA);
  ac2 = bmp085ReadInt(0xAC);
  ac3 = bmp085ReadInt(0xAE);
  ac4 = bmp085ReadInt(0xB0);
  ac5 = bmp085ReadInt(0xB2);
  ac6 = bmp085ReadInt(0xB4);
```

```
  b1 = bmp085ReadInt(0xB6);
  b2 = bmp085ReadInt(0xB8);
  mb = bmp085ReadInt(0xBA);
  mc = bmp085ReadInt(0xBC);
  md = bmp085ReadInt(0xBE);
}

long bmp085GetPressure(unsigned long up)//////////////// presure getting function for BMP085 sensor
{
  long x1, x2, x3, b3, b6, p;
  unsigned long b4, b7;
  b6 = b5 - 4000;
  x1 = (b2 * (b6 * b6)>>12)>>11;
  x2 = (ac2 * b6)>>11;
  x3 = x1 + x2;
  b3 = (((((long)ac1)*4 + x3)<<OSS) + 2)>>2;
  x1 = (ac3 * b6)>>13;
  x2 = (b1 * ((b6 * b6)>>12))>>16;
  x3 = ((x1 + x2) + 2)>>2;
  b4 = (ac4 * (unsigned long)(x3 + 32768))>>15;
  b7 = ((unsigned long)(up - b3) * (50000>>OSS));
  if (b7 < 0x80000000)
    p = (b7<<1)/b4;
  else
    p = (b7/b4)<<1;
  x1 = (p>>8) * (p>>8);
  x1 = (x1 * 3038)>>16;
  x2 = (-7357 * p)>>16;
  p += (x1 + x2 + 3791)>>4;
```

```
  return p;
}
char bmp085Read(unsigned char address)
{
  unsigned char data;
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(address);
  Wire.endTransmission();
  Wire.requestFrom(BMP085_ADDRESS, 1);
  while(!Wire.available())
    ;
  return Wire.read();
}
int bmp085ReadInt(unsigned char address)
{
  unsigned char msb, lsb;
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(address);
  Wire.endTransmission();
  Wire.requestFrom(BMP085_ADDRESS, 2);
  while(Wire.available()<2)  ;
  msb = Wire.read();
  lsb = Wire.read();
  return (int) msb<<8 | lsb;
}
unsigned int bmp085ReadUT()
{
  unsigned int ut;
  Wire.beginTransmission(BMP085_ADDRESS);
```

```
  Wire.write(0xF4);
  Wire.write(0x2E);
  Wire.endTransmission();
  delay(5);
  ut = bmp085ReadInt(0xF6);
  return ut;
}
unsigned long bmp085ReadUP()
{
  unsigned char msb, lsb, xlsb;
  unsigned long up = 0;
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(0xF4);
  Wire.write(0x34 + (OSS<<6));
  Wire.endTransmission();
  delay(2 + (3<<OSS));
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(0xF6);
  Wire.endTransmission();
  Wire.requestFrom(BMP085_ADDRESS, 3);
  while(Wire.available() < 3);
  msb = Wire.read();
  lsb = Wire.read();
  xlsb = Wire.read();
  up = (((unsigned long) msb << 16) | ((unsigned long) lsb << 8) | (unsigned long) xlsb) >> (8-OSS);
  return up;
}
boolean readSensor(int * temperature_decidegrees_c, int * rel_humidity_decipercent) {
  initState();
```

```
  attachInterrupt(sensorPin - 2, sensorLineChange, CHANGE);
  if (!requestSensorRead()) {
    flagError();
  } else {
    const long startMillis = millis();
    do {
      delay(100);
    } while ( signalLineChanges < 86 && ( (millis() - startMillis) < sensorReadTimeoutMillis));
    detachInterrupt(sensorPin - 2);
    if (signalLineChanges != 86) {
      sprintf(errorMsgBuf, "*** MISSED INTERRUPTS ***: Expected 86 line changes, saw %i", signalLineChanges);
      flagError();
    } else {
      analyseTimings(temperature_decidegrees_c, rel_humidity_decipercent);
    }
  }
  return !errorFlag;
}
void sensorLineChange() {
  const long pulseMicros = micros() - lastTransitionMicros;
  lastTransitionMicros = micros();
  timings[signalLineChanges] = pulseMicros;
  signalLineChanges++;
}
void initState() {
  detachInterrupt(sensorPin - 2);
  for (int i = 0; i < 86; i++) { timings[i] = 0; }
  errorFlag = false;
  lastTransitionMicros = micros();
```

```
  signalLineChanges = 0;
}
void debugPrintTimings() {
  for (int i = 0; i < 86; i++) {
    if (i%10==0) { Serial.print("\n\t"); }
    char buf[24];
    sprintf(buf, i%2==0 ? "H[%02i]: %-3i  " : "L[%02i]: %-3i  ", i, timings[i]);
    Serial.print(buf);
  }
  Serial.print("\n");
}
void analyseTimings(int * temperature_decidegrees_c, int * rel_humidity_decipercent) {
  int timingsIdx = 5;
  int humid16 = readnbits(&timingsIdx, 16);
  if (errorFlag) {
    Serial.println("Failed to capture humidity data");
    return;
  }
  int temp16 = readnbits(&timingsIdx, 16);
  if (errorFlag) {
    Serial.println("Failed to capture temperature data");
    return;
  }
  int checksum8 = readnbits(&timingsIdx, 8);
  if (errorFlag) {
    Serial.println("Failed to capture checksum");
    return;
  }
  byte cs = (byte)(humid16>>8) + (byte)(humid16&0xFF) + (byte)(temp16>>8) + (byte)(temp16&0xFF);
```

```
  if (cs != checksum8) {
    sprintf(errorMsgBuf, "Checksum mismatch, bad sensor read");
    flagError();
  }
  if ( temp16 & (1<<15) ) {
    temp16 = -(temp16 & (~(1<<15)));
  }
  if (!errorFlag) {
    *temperature_decidegrees_c = temp16;
    *rel_humidity_decipercent = humid16;
  }
}
int readnbits(int * timingsIdx, int nbits) {
  const int * t = timings + *timingsIdx;
  const int * tStop = t + nbits*2;
  int result = 0;
  char buf[12];
  while (t != tStop) {
    checkPreBitLowPulse( *(t++), (*timingsIdx)++ );
    result = shiftNextBit( result, *(t++), (*timingsIdx)++ );
  }
  return result;
}
int shiftNextBit(int oldValue, int pulseMicros, int timingIndex) {
  if (pulseMicros > 10 && pulseMicros < 40) {
    return (oldValue<<1) | 0;
  } else if (pulseMicros > 60 && pulseMicros < 85) {
    return (oldValue<<1) | 1;
  } else {
```

```
    sprintf(errorMsgBuf, "Bad bit pulse length: %i us at timing idx %i", pulseMicros, timingIndex);
    flagError();
    return 0xFFFFFFFF;
  }
}
void checkPreBitLowPulse(int pulseMicros, int timingIndex) {
  if (pulseMicros <= 35 || pulseMicros >= 75) {
    sprintf(errorMsgBuf, "Low pulse before bit transmit (%i us) outside 45-70us tolerance at timing idx %i", pulseMicros,
timingIndex);
    flagError();
  }
}
boolean requestSensorRead() {
  if (digitalRead(sensorPin) != HIGH) {
    sprintf(errorMsgBuf, "Line not HIGH at entry to requestSensorRead()");
    flagError();
    return false;
  }
  digitalWrite(sensorPin, LOW);
  pinMode(sensorPin, OUTPUT);
  delayMicroseconds(7000);
  digitalWrite(sensorPin, HIGH);
  delayMicroseconds(30);
  pinMode(sensorPin, INPUT);

  int pulseLength = pulseIn(sensorPin, LOW, 200);
  if (pulseLength == 0) {
    sprintf(errorMsgBuf, "Sensor read failed: Sensor never pulled line LOW after initial request");
    flagError();
```

```
    return false;
  }
  delayMicroseconds(5);
  pulseLength = pulseIn(sensorPin, HIGH, 200);
  if (pulseLength == 0) {
    sprintf(errorMsgBuf, "Sensor read failed: Sensor didn't go back HIGH after LOW response to read request");
    flagError();
    return false;
  }
  return true;
}
void flagError() {
  pinMode(sensorPin, INPUT);
  digitalWrite(sensorPin, HIGH);
  errorFlag = true;
  Serial.println(errorMsgBuf);
}
void speedcalculation(){
  windspeed=(2*3.14*0.25*Rev)*3.6;
  //Serial.print(windspeed);
  //Serial.print(Rev);
   Rev=0;
  //Serial.print("\n");
}
void windcount()
 {
   Rev=Rev+1;
 }
void setup()
```

```
{
  cs_4_6.set_CS_AutocaL_Millis(0xFFFFFFFF);
  Serial.begin(9600);
  Wire.begin();
  cell.begin(9600);
  delay(35000); // give the GSM module time to initialise, locate network etc.
  bmp085Calibration();
  pinMode(sensorPin, INPUT);
  digitalWrite(sensorPin, HIGH);
  delay(5000);
  //Serial.begin(serialBaudRate);
  attachInterrupt(1, windcount, FALLING);
  Alarm.timerRepeat(60, speedcalculation);
  delay(sensorReadIntervalMs);
}
void loop()
{
//long start = millis();
long CapSen =  cs_4_6.capacitiveSensorRaw(1023);
pressure = bmp085GetPressure(bmp085ReadUP());
altitude = (float)44330 * (1 - pow(((float) pressure/p0), 0.190295));
int temperature;
int humidity;
boolean success = readSensor(&temperature, &humidity);
 if (success) {
    char buf[128];
    sprintf(buf, "Temperature %i.%i degrees C at  %i.%i relative humidity", temperature/10, abs(temperature%10),
humidity/10, humidity%10);
    Serial.println(buf);
```

```
  //globalbuf[128]=buf[128];
   sensors[0]=buf[128];

 }
 Serial.print("Pressure: ");
 Serial.print(pressure, DEC);
 Serial.println(" Pa");
 Serial.print("Altitude: ");
 Serial.print(altitude, 2);
 Serial.println(" m");
 Alarm.delay(200); // wait one second between clock display
 //Serial.println("CapSen");
 //Serial.println(CapSen);
 WaterLevel=((CapSen-21593.6)/(72.66)); // for water level calculations
Serial.print("WaterLevel: ");
Serial.print(WaterLevel);
Serial.print(" ml");
Serial.println();
Serial.print("windspeed: ");
Serial.print(windspeed);
Serial.print(" km/h");
Serial.println();
sensors[1]=pressure;
sensors[2]=altitude;
sensors[3]=WaterLevel;
sensors[4]=windspeed;
cell.println("AT");
delay(200);
cell.println("AT+CMGF=1"); // set SMS mode to text
```

```
delay(200);
cell.print("AT+CMGS=");  // now send message...
cell.write((byte)34); // ASCII equivalent of "
cell.print(phoneNumber);
cell.write((byte)34);  // ASCII equivalent of "
cell.println();
delay(200); // give the module some thinking time
cell.print("Amount Of Rainfall =");
cell.print(sensors[3]);
cell.print("ml");
cell.println();
cell.print("Windspeed =");
cell.print(sensors[4]);
cell.print("Km/hour");
cell.println();
cell.print(sensors[0]);
cell.println();
cell.print("Pressure =");
cell.print(sensors[1]);
cell.print("Pa");
cell.println();
cell.write((byte)26);  // ASCII equivalent of Ctrl-Z
cell.println();
delay(15000); // the SMS module needs time to return to OK status
if(WaterLevel>200){
digitalWrite(pum,HIGH);
delay(20000);
digitalWrite(pum,LOW);
}
```

References

[1] http://en.wikipedia.org/wiki/Automatic_weather_station

[2] http://www.electronics-tutorials.ws/waveforms/555_oscillator.html

[3] http://www.circuitstoday.com/frequency-to-voltage-converter-using-lm331

[4] http://electrotech.ps/os/product_info.php?products_id=119

[5] http://www.alldatasheet.com/datasheet-pdf/pdf/5178/MOTOROLA/MPX4115.html

[6] http://microcontrollershop.com/product_info.php?products_id=2125

[7] http://www.engineersgarage.com

[8] http://arduino.cc/en/Main/ArduinoBoardUno

[9] https://www.sparkfun.com/products/9607

[10] https://www.sparkfun.com/products/10167