PLEASE NOTE: THE MATTER

PRODUCED BELOW IS

REFERENCE MATERIAL ONLY

FOR PREPARING THE

PROJECT REPORT. IT IS NOT

THE COMPLETE PROJECT

REPORT.

A PROJECT REPORT ON

Submitted in partial fulfillment of the requirements

For the award of the degree

BACHELOR OF ENGINEERING

IN

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ENGINEERING

SUBMITTED BY

-------------------- (--------------)

--------------------- (---------------)

--------------------- (---------------)

DEPARTMENT OF \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ENGINEERING

\_\_\_\_\_\_\_\_\_\_COLLEGE OF ENGINEERING

AFFILIATED TO \_\_\_\_\_\_\_\_\_\_\_ UNIVERSITY

**CERTIFICATE**

This is to certify that the dissertation work entitled

“**Weather Imaging CubeSat with Telemetry Transmission”**

is the work done by\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ submitted in partial fulfillment for the award of ‘BACHELOR OF ENGINEERING (B.E)’ in Electronics and Communication Engineering from\_\_\_\_\_\_\_ College of Engineering affiliated to \_\_\_\_\_\_\_\_\_ University , Hyderabad .

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_**

**(Head of the department, ECE) (Assistant Professor)**

**EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mentioning of the people whose constant guidance and encouragement made it possible. We take pleasure in presenting before you, our project, which is result of studied blend of both research and knowledge.

We express our earnest gratitude to our internal guide, Assistant Professor \_\_\_\_\_\_\_\_\_\_\_\_\_\_, Department of ECE, our project guide, for his constant support, encouragement and guidance. We are grateful for his cooperation and his valuable suggestions.

Finally, we express our gratitude to all other members who are involved either directly or indirectly for the completion of this project.

**DECLARATION**

We, the undersigned, declare that the project entitled

‘**Weather Imaging CubeSat with Telemetry Transmission**, being submitted in partial fulfillment for the award of Bachelor of Engineering Degree in Electronics and Communication Engineering, affiliated to \_\_\_\_\_\_\_\_\_ University, is the work carried out by us.

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

**ABSTRACT**

Satellites are launched in orbit for a variety of purposes including communication, GPS, Weather imaging and similar applications. Weather imaging satellites are used to transmit data about weather parameters that can be used for prediction and forecasting systems.

So here we develop a demo weather imaging and parameter transmission satellite to transmit weather data for analysis back to the earth. The satellite delivers following functions:

* Record and transmit earth weather imaging data
* Sense and transmit infrared radiation levels to detect solar winds
* Sense and transmit temperature
* Self Charge itself using Solar Power

The system makes use of an STM32 controller along with a solar panel, battery for power supply, a magnetometer, infrared sensor, temperature sensor, a camera and 2.4Ghz rf transmitter to develop the satellite. We here develop a basic Cubesat design without an ACDS stabilizer system with more focus towards weather data gathering and transmission part.

The CUBE SAT body is a cubicle shaped frame made of 4 sides. The 4 parts are merged to develop a cube with mounting holes for the controller, sensors, circuit board, solar panels and battery. The solar panels are used to generate energy that is storec in battery using charge controller and charging circuitry.

The battery charging and auto cutoff is managed by circuitry to avoid overcharging and maintain battery health. The magnetometer is used to check orientation of the cubesat. The sensor transmits orientation data to controller. The Temperature sensor is used to measure the orbital temperature. Infrared sensor is mounted on top to help measure the infrared radiation from the sun and detect solar waves/blasts.

The camera is mounted at the bottom of the cubesat for capturing live footage of the earth from above. This footage along with the sensor data is transmitted by the controller through a 2.4Ghz RF transmitter module with a high gain antenna for max distance transmission. The sensor data is also displayed on an LCD Display for reference.

The transmitted data is now received by a receiving station on the ground. The ground station circuit consists of an LCD display and footage display. The 2.4Ghz receiver is used to receive the signals from cubesat and display all the data for further processing.

**INTRODUCTION TO EMBEDDED SYSTEMS**

**What is embedded system?**

An Embedded System is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function. An embedded system is a microcontroller-based, software driven, reliable, real-time control system, autonomous, or human or network interactive, operating on diverse physical variables and in diverse environments and sold into a competitive and cost conscious market.

An embedded system is not a computer system that is used primarily for processing, not a software system on PC or UNIX, not a traditional business or scientific application. High-end embedded & lower end embedded systems. High-end embedded system - Generally 32, 64 Bit Controllers used with OS. Examples Personal Digital Assistant and Mobile phones etc .Lower end embedded systems - Generally 8,16 Bit Controllers used with an minimal operating systems and hardware layout designed for the specific purpose.

* **SYSTEM DESIGN CALLS:**

**EMBEDDED SYSTEM DESIGN CYCLE**

**Characteristics of Embedded System**

An embedded system is any computer system hidden inside a product other than a computer.

They will encounter a number of difficulties when writing embedded system software in addition to those we encounter when we write applications.

* + Throughput – Our system may need to handle a lot of data in a short period of time.
  + Response–Our system may need to react to events quickly.
  + Testability–Setting up equipment to test embedded software can be difficult.
  + Debugability–Without a screen or a keyboard, finding out what the software is doing wrong (other than not working) is a troublesome problem.
  + Reliability – embedded systems must be able to handle any situation without human intervention.
  + Memory space – Memory is limited on embedded systems, and you must make the software and the data fit into whatever memory exists.
  + Program installation – you will need special tools to get your software into embedded systems.
  + Power consumption – Portable systems must run on battery power, and the software in these systems must conserve power.
  + Processor hogs – computing that requires large amounts of CPU time can complicate the response problem.
  + Cost – Reducing the cost of the hardware is a concern in many embedded system projects; software often operates on hardware that is barely adequate for the job.
* Embedded systems have a microprocessor/ microcontroller and a memory. Some have a serial port or a network connection. They usually do not have keyboards, screens or disk drives.

**APPLICATIONS**

1. Military and aerospace embedded software applications
2. Communication Applications
3. Industrial automation and process control software
4. Mastering the complexity of applications.
5. Reduction of product design time.
6. Real time processing of ever increasing amounts of data.
7. Intelligent, autonomous sensors.

**CLASSIFICATION**

* Real Time Systems.
* RTS is one which has to respond to events within a specified deadline.
* A right answer after the dead line is a wrong answer.

**RTS CLASSIFICATION**

* Hard Real Time Systems
* Soft Real Time System

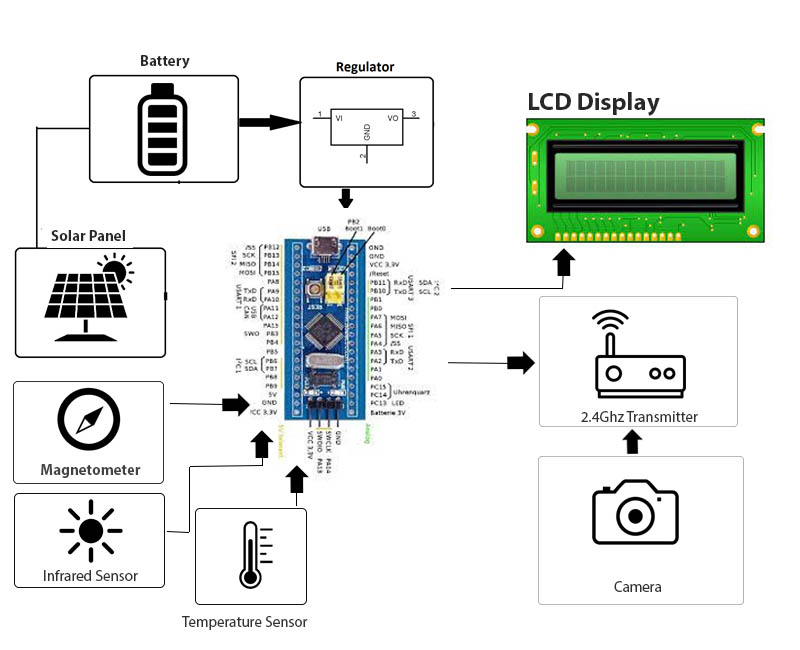
**HARD REAL TIME SYSTEM**

* "Hard" real-time systems have very narrow response time.
* Example: Nuclear power system, Cardiac pacemaker.

**SOFT REAL TIME SYSTEM**

* "Soft" real-time systems have reduced constrains on "lateness" but still must operate very quickly and repeatable.
* Example: Railway reservation system – takes a few extra seconds the data remains valid.

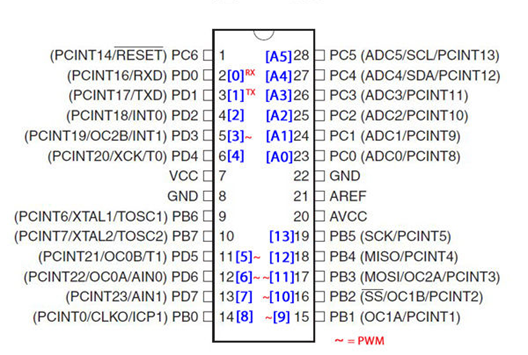
**BLOCK DIAGRAM**



**HARDWARE COMPONENTS**

* Stm32 Controller
* Atmega328
* Battery
* Solar Panels
* Camera Module
* Magnetometer
* Infrared Sensor
* Temperature Sensor
* OLED
* Buzzer
* LED’s
* PCB Board
* Resistors
* Capacitors
* Transistors
* Cables and Connectors
* Cubesat Frame
* Nuts and Bolts

**ATMEGA328**



**Introduction:**

The Atmel ATmega328P is a 32K 8-bit microcontroller based on the AVR architecture. Many instructions are executed in a single clock cycle providing a throughput of almost 20 MIPS at 20MHz. The ATMEGA328-PU comes in an PDIP 28 pin package and is suitable for use on our [28 pin AVR Development Board](http://www.protostack.com/product_by_model.php?model=PB-MC-AVR28).

The computer on one hand is designed to perform all the general purpose tasks on a single machine like you can use a computer to run a software to perform calculations or you can use a computer to store some multimedia file or to access internet through the browser, whereas the microcontrollers are meant to perform only the specific tasks, for e.g., switching the AC off automatically when room temperature drops to a certain defined limit and again turning it ON when temperature rises above the defined limit.

There are number of popular families of microcontrollers which are used in different applications as per their capability and feasibility to perform the desired task, most common of

these are 8051, AVR and PIC microcontrollers. In this we will introduce you with AVR family of microcontrollers.

**Features include:**

* High Performance, Low Power Design
* 8-Bit Microcontroller Atmel® AVR® advanced RISC architecture
  + 131 Instructions most of which are executed in a single clock cycle
  + Up to 20 MIPS throughput at 20 MHz
  + 32 x 8 working registers
  + 2 cycle multiplier
* Memory Includes
  + 32KB of of programmable FLASH
  + 1KB of EEPROM
  + 2KB SRAM
  + 10,000 Write and Erase Cycles for Flash and 100,000 for EEPROM
  + Data retention for 20 years at 85°C and 100 years at 25°C
  + Optional boot loader with lock bits
    - In System Programming (ISP) by via boot loader
    - True Read-While-Write operation
  + Programming lock available for software security
* Features Include
  + 2 x 8-bit Timers/Counters each with independent prescaler and compare modes
  + A single 16-bit Timer/Counter with an idependentprescaler, compare and capture modes
  + Real time counter with independent oscillator
  + 10 bit, 6 channel analog to digital Converter
  + 6 pulse width modulation channels
  + Internal temperature sensor
  + Serial USART (Programmable)
  + Master/Slave SPI Serial Interface - (Philips I2C compatible)
  + Programmable watchdog timer with independent internal oscillator
  + Internal analog comparator
  + Interrupt and wake up on pin change
* Additional Features Features
  + Internal calibrated oscillator
  + Power on reset and programmable brown out detection
  + External and internal interrupts
* 6 sleep modes including idle, ADC noise reduction, power save, power down, standby, and extended standby
* I/O and Package
  + 23 programmable I/O lines
  + 28 pin PDIP package
* Operating voltage:
  + 1.8 - 5.5V
* Operating temperature range:
  + 40°C to 85°C
* Speed Grades:
  + 0-4 MHz at 1.8-5.5V
  + 0-10 MHz at 2.7-5.5V
  + 0-20 MHz at 4.5-5.5V
* Low power consumption mode at 1.8V, 1 MHz and 25°C:
  + Active Mode: 0.3 mA
  + Power-down Mode: 0.1 μA
  + Power-save Mode: 0.8 μA (Including 32 kHz RTC)

|  |  |
| --- | --- |
| Flash: | 32 KBytes |
| EEPROM: | 1 KBytes |
| SRAM: | 2 KBytes |
| Max I/O Pins: | 23 |
| Frequency Max: | 20 MHz |
| VCC: | 1.8-5.5 |
| 10-bit A/D Channels: | 6 |
| Analog Comparator: | Yes |
| 16-bit Timers: | 1 |
| 8-bit Timer: | 2 |
| Brown Out Detector: | Yes |
| Ext Interrupts: | 2 |
| Hardware Multiplier: | Yes |
| Interrupts: | 26 |
| ISP: | Yes |
| On Chip Oscillator: | Yes |
| PWM Channels: | 6 |
| RTC: | Yes |
| Self Program Memory: | Yes |
|  |  |
| SPI: | 1 |
| TWI: | Yes |
| UART: | 1 |
| Watchdog: | Yes |
| Pacakage: | Lead Free PDIP 28 |

In our days, there have been many advancement in the field of Electronics and many cutting edge technologies are being  developed every day, but still 8 bit microcontrollers have its own role in the digital electronics market dominated by 16-32 & 64 bit digital devices. Although powerful microcontrollers with higher processing capabilities exist in the market, 8bit microcontrollers still hold its value because of their easy-to-understand-operation, very much high popularity, ability to simplify a digital circuit, low cost compared to features offered, addition of many new features in a single IC and interest of manufacturers and consumers.

Today’s microcontrollers are much different from what it were in the initial stage, and the number of manufacturers are much more in count than it was a decade or two ago. At present some of the major manufacturers are Microchip (publication: PIC microcontrollers), Atmel (publication: AVR microcontrollers), Hitachi, Phillips, Maxim, NXP, Intel etc.  Our interest is upon ATmega32. It belongs to Atmel’s AVR series micro controller family. Let’s see the features.

**PIN count:** Atmega32 has got 40 pins. Two for Power (pin no.10: +5v, pin no. 11: ground), two for oscillator (pin 12, 13), one for reset (pin 9), three for providing necessary power and reference voltage to its internal ADC, and 32 (4×8) I/O pins.

**About I/O pins:** ATmega32 is capable of handling analogue inputs. Port A can be used as either DIGITAL I/O Lines or each individual pin can be used as a single input channel to the internal ADC of ATmega32, plus a pair of pins AREF, AVCC & GND (refer to [ATmega32 datasheet](http://www.atmel.com/Images/doc2503.pdf)) together can make an ADC channel.

No pins can perform and serve for two purposes (for an example: Port A pins cannot work as a Digital I/O pin while the Internal ADC is activated) at the same time. It’s the programmers responsibility to resolve the conflict in the circuitry and the program. Programmers are advised to have a look to the priority tables and the internal configuration from the datasheet.

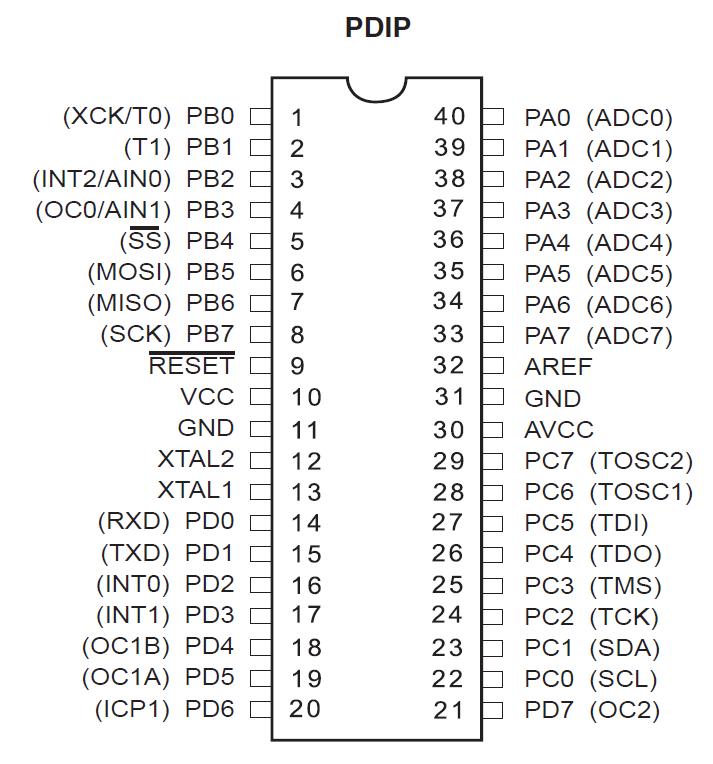
**Digital I/O pins:** ATmega32 has 32 pins (4portsx8pins) configurable as Digital I/O pins.

**Timers:** 3 Inbuilt timer/counters, two 8 bit (timer0, timer2) and one 16 bit (timer1).

**ADC:** It has one successive approximation type ADC in which total 8 single channels are selectable. They can also be used as 7 (for TQFP packages) or 2 (for DIP packages) differential channels. Reference is selectable, either an external reference can be used or the internal 2.56V reference can be brought into action.  There external reference can be connected to the AREF pin.

**Communication Options:**  ATmega32 has three data transfer modules embedded in it. They are

* Two  Wire Interface
* USART
* Serial Peripheral Interface



**Analog comparator:**  On-chip analog comparator is available. An interrupt is assigned for different comparison result obtained from the inputs.

**External Interrupt:** 3External interrupt is accepted. Interrupt sense is configurable.

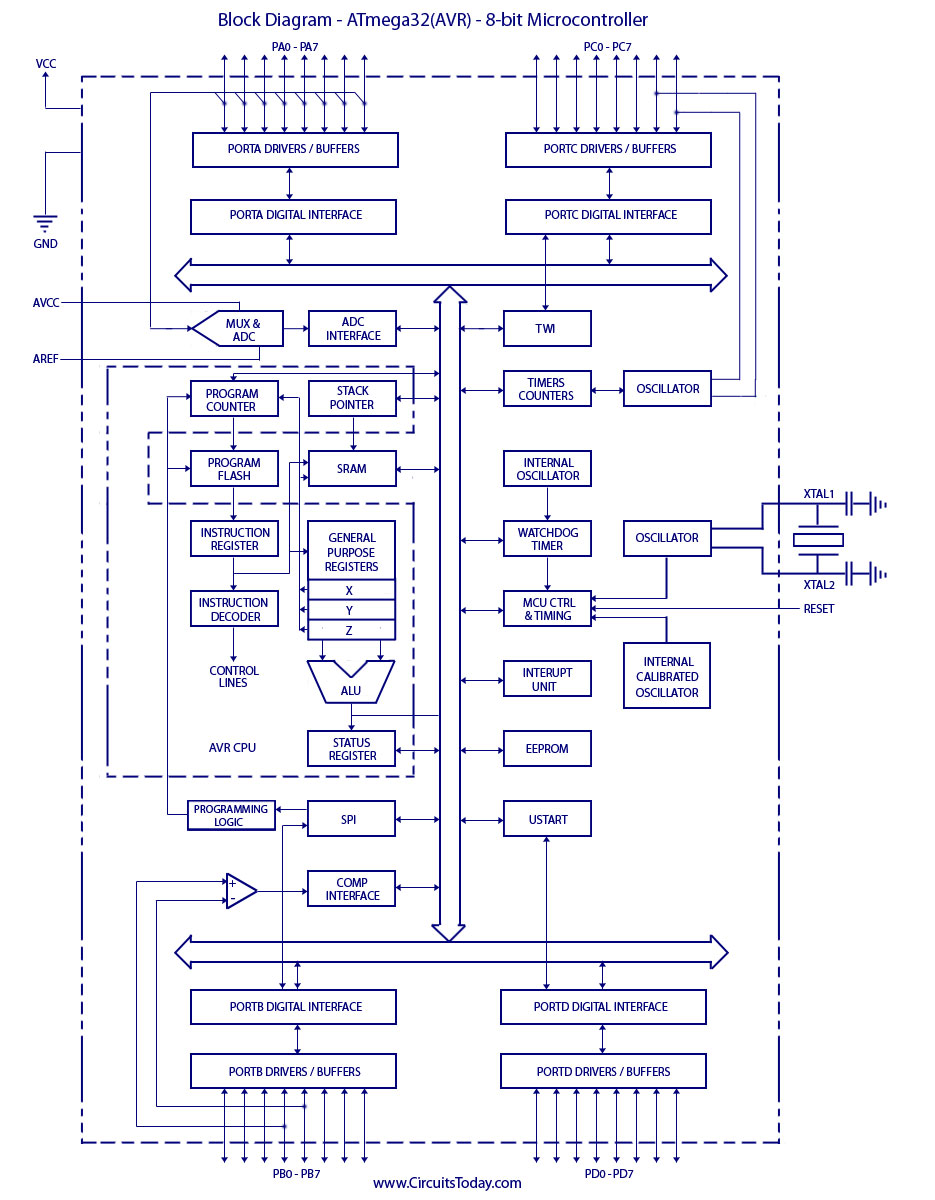
**Memory:**  It has 32Kbytes of In-System Self-programmable Flash program memory, 1024 Bytes EEPROM, 2Kbytes Internal SRAM. Write/Erase Cycles: 10,000 Flash / 100,000 EEPROM.

**Clock:** It can run at a frequency from 1 to 16 MHz. Frequency can be obtained from external Quartz Crystal, Ceramic crystal or an R-C network. Internal calibrated RC oscillator can also be used.

**More Features**: Up to 16 MIPS throughput at 16MHz. Most of the instruction executes in a single cycle. Two cycle on-chip multiplication. 32 × 8 General Purpose Working Registers

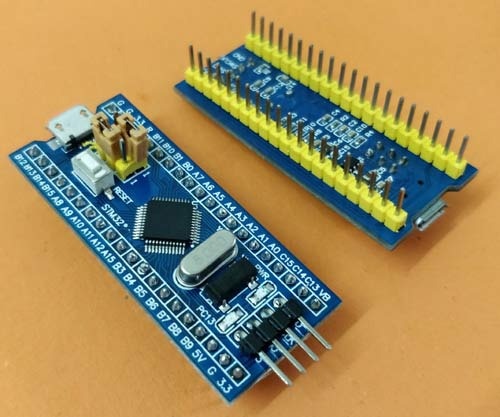
**Debug:** JTAG boundary scan facilitates on chip debug.

**Programming:** Atmega32 can be programmed either by In-System Programming via Serial peripheral interface or by Parallel programming. Programming via JTAG interface is also possible. Programmer must ensure that SPI programming and JTAG are not be disabled using  fuse bits; if the programming is supposed to be done using SPI or JTAG.



**STM32 CONTROLLER**

The STM32F103C8T6 (also known as ‘STM32’ or ‘Blue Pill”) is a cheap development board based on the ARM Cortex M3 microprocessor.



**Naming Convention of STM microcontrollers**

Parameter Meaning

STM name of the manufacturer (STMicroelectronics)

32 32 bit ARM architecture

F Foundation

1 Core (ARM Cortex M3)

03 Line (describes peripherals and speed)

C 48 pins

8 64 KB flash memory

T LQFP package (Low Profile Quad Flat Pack)

6 Operating Temperature Range (-40 °C to 85 °C)

**Technical Specifications of STM32**

Parameter Meaning

Architecture 32 bit ARM Cortex M3

Operating Voltage 2.7V to 3.6V

CPU Frequency 72 MHz

Number of GPIO pins 37

Number of PWM pins 12

Analog Input Pins 10 (12 bit resolution)

I2C Peripherals 2

SPI Peripherals 2

CAN 2.0 Peripheral 1

Timers 3(16-bit), 1

Flash Memory 64KB

RAM 20kB

**HC12**

The HC-12 is a half-duplex wireless serial communication module with 100 channels in the 433.4-473.0 MHz range that is capable of transmitting up to 1 km. This project will begin by using the HC-12 to create a wireless link between two computers and end with a second article that creates a simple wireless GPS tracker.The HC-12 is a half-duplex 20 dBm (100 mW) transmitter paired with a receiver that has -117 dBm (2×10-15 W) sensitivity at 5000 bps.Paired with an external antenna, these transceivers are capable of communicating up to and possibly slightly beyond 1 km in the open and are more than adequate for providing coverage throughout a typical house.



**BUZZER**

An audio signaling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.

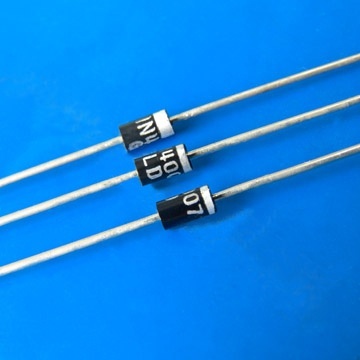
****

The pin configuration of the buzzer is shown below. It includes two pins namely positive and negative. The positive terminal of this is represented with the ‘+’ symbol or a longer terminal. This terminal is powered through 6Volts whereas the negative terminal is represented with the ‘-‘symbol or short terminal and it is connected to the GND terminal.

**1N4007**

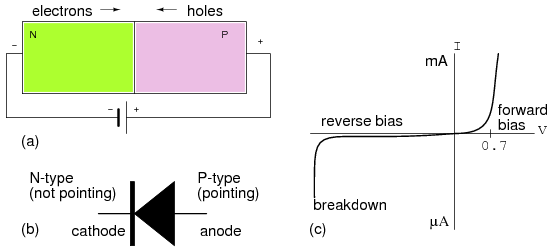
Diodes are used to convert AC into DC these are used as half wave rectifier or full wave rectifier. Three points must he kept in mind while using any type of diode.

1. Maximum forward current capacity
2. Maximum reverse voltage capacity
3. Maximum forward voltage capacity



The number and voltage capacity of some of the important diodes available in the market are as follows:

* Diodes of number IN4001, IN4002, IN4003, IN4004, IN4005, IN4006 and IN4007 have maximum reverse bias voltage capacity of 50V and maximum forward current capacity of 1 Amp.
* Diode of same capacities can be used in place of one another. Besides this diode of more capacity can be used in place of diode of low capacity but diode of low capacity cannot be used in place of diode of high capacity. For example, in place of IN4002; IN4001 or IN4007 can be used but IN4001 or IN4002 cannot be used in place of IN4007.The diode BY125made by company BEL is equivalent of diode from IN4001 to IN4003. BY 126 is equivalent to diodes IN4004 to 4006 and BY 127 is equivalent to diode IN4007.



PN JUNCTION OPERATION

Now that you are familiar with P- and N-type materials, how these materials are joined together toform a diode, and the function of the diode, let us continue our discussion with the operation of the PNjunction. But before we can understand how the PN junction works, we must first consider current flow inthe materials that make up the junction and what happens initially within the junction when these twomaterials are joined together.

Current Flow in the N-Type Material

Conduction in the N-type semiconductor, or crystal, is similar to conduction in a copper wire. Thatis, with voltage applied across the material, electrons will move through the crystal just as current wouldflow in a copper wire. This is shown in figure 1-15. The positive potential of the battery will attract thefree electrons in the crystal. These electrons will leave the crystal and flow into the positive terminal ofthe battery. As an electron leaves the crystal, an electron from the negative terminal of the battery willenter the crystal, thus completing the current path. Therefore, the majority current carriers in the N-typematerial (electrons) are repelled by the negative side of the battery and move through the crystal towardthe positive side of the battery.

Current Flow in the P-Type Material

Current flow through the P-type material is illustrated. Conduction in the P material isby positive holes, instead of negative electrons. A hole moves from the positive terminal of the P materialto the negative terminal. Electrons from the external circuit enter the negative terminal of the material andfill holes in the vicinity of this terminal. At the positive terminal, electrons are removed from the covalentbonds, thus creating new holes. This process continues as the steady stream of holes (hole current) movestoward the negative terminal

**RESISTORS**

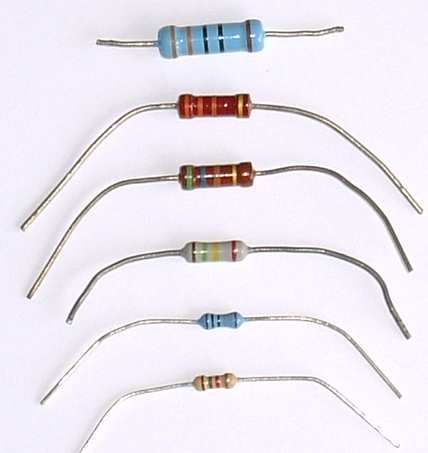
A resistor is a two-terminal electronic component designed to oppose an electric current by producing a voltage drop between its terminals in proportion to the current, that is, in accordance with Ohm's law:

*V* = *IR*

Resistors are used as part of electrical networks and electronic circuits. They are extremely commonplace in most electronic equipment. Practical resistors can be made of various compounds and films, as well as resistance wire (wire made of a high-resistivity alloy, such as nickel/chrome).

The primary characteristics of resistors are their resistance and the power they can dissipate. Other characteristics include temperature coefficient, noise, and inductance. Less well-known is critical resistance, the value below which power dissipation limits the maximum permitted current flow, and above which the limit is applied voltage. Critical resistance depends upon the materials constituting the resistor as well as its physical dimensions; it's determined by design.

Resistors can be integrated into hybrid and printed circuits, as well as integrated circuits. Size, and position of leads (or terminals) are relevant to equipment designers; resistors must be physically large enough not to overheat when dissipating their power.



A resistor is a two-[terminal](http://en.wikipedia.org/wiki/Terminal_%28electronics%29" \o "Terminal (electronics))[passive](http://en.wikipedia.org/wiki/Passivity_%28engineering%29)[electronic component](http://en.wikipedia.org/wiki/Electronic_component) which implements [electrical resistance](http://en.wikipedia.org/wiki/Electrical_resistance) as a circuit element. When a voltage V is applied across the terminals of a resistor, a current I will flow through the resistor in [direct proportion](http://en.wikipedia.org/wiki/Direct_proportion) to that voltage. The reciprocal of the constant of proportionality is known as the [resistance](http://en.wikipedia.org/wiki/Resistance) R, since, with a given voltage V, a larger value of R further "resists" the flow of current I as given by [Ohm's law](http://en.wikipedia.org/wiki/Ohm%27s_law):

I = {V \over R}

Resistors are common elements of [electrical networks](http://en.wikipedia.org/wiki/Electrical_networks) and electronic circuits and are ubiquitous in most electronic equipment. Practical resistors can be made of various compounds and films, as well as [resistance wire](http://en.wikipedia.org/wiki/Resistance_wire) (wire made of a high-resistivity alloy, such as nickel-chrome). Resistors are also implemented within [integrated circuits](http://en.wikipedia.org/wiki/Integrated_circuits), particularly analog devices, and can also be integrated into [hybrid](http://en.wikipedia.org/wiki/Hybrid_circuit) and [printed circuits](http://en.wikipedia.org/wiki/Printed_circuit_board).

The electrical functionality of a resistor is specified by its resistance: common commercial resistors are manufactured over a range of more than 9 [orders of magnitude](http://en.wikipedia.org/wiki/Orders_of_magnitude). When specifying that resistance in an electronic design, the required precision of the resistance may require attention to the [manufacturing tolerance](http://en.wikipedia.org/wiki/Engineering_tolerance#Electrical_component_tolerance) of the chosen resistor, according to its specific application. The [temperature coefficient](http://en.wikipedia.org/wiki/Temperature_coefficient) of the resistance may also be of concern in some precision applications. Practical resistors are also specified as having a maximum [power](http://en.wikipedia.org/wiki/Power_%28physics%29) rating which must exceed the anticipated power dissipation of that resistor in a particular circuit: this is mainly of concern in power electronics applications. Resistors with higher power ratings are physically larger and may require [heat sinking](http://en.wikipedia.org/wiki/Heat_sink). In a high voltage circuit, attention must sometimes be paid to the rated maximum working voltage of the resistor.

The series [inductance](http://en.wikipedia.org/wiki/Inductance) of a practical resistor causes its behavior to depart from ohms law; this specification can be important in some high-frequency applications for smaller values of resistance. In a [low-noise amplifier](http://en.wikipedia.org/wiki/Low-noise_amplifier) or [pre-amp](http://en.wikipedia.org/wiki/Pre-amp) the noise characteristics of a resistor may be an issue. The unwanted inductance, excess noise, and temperature coefficient are mainly dependent on the technology used in manufacturing the resistor. They are not normally specified individually for a particular family of resistors manufactured using a particular technology. A family of discrete resistors is also characterized according to its form factor, that is, the size of the device and position of its leads (or terminals) which is relevant in the practical manufacturing of circuits using them.

## Units

The [ohm](http://en.wikipedia.org/wiki/Ohm_%28unit%29) (symbol: [Ω](http://en.wikipedia.org/wiki/%CE%A9)) is the [SI](http://en.wikipedia.org/wiki/SI) unit of [electrical resistance](http://en.wikipedia.org/wiki/Electrical_resistance), named after [Georg Simon Ohm](http://en.wikipedia.org/wiki/Georg_Simon_Ohm). An ohm is equivalent to a [volt](http://en.wikipedia.org/wiki/Volt) per [ampere](http://en.wikipedia.org/wiki/Ampere). Since resistors are specified and manufactured over a very large range of values, the derived units of milliohm (1 mΩ = 10−3 Ω), kilohm (1 kΩ = 103 Ω), and megohm (1 MΩ = 106 Ω) are also in common usage.

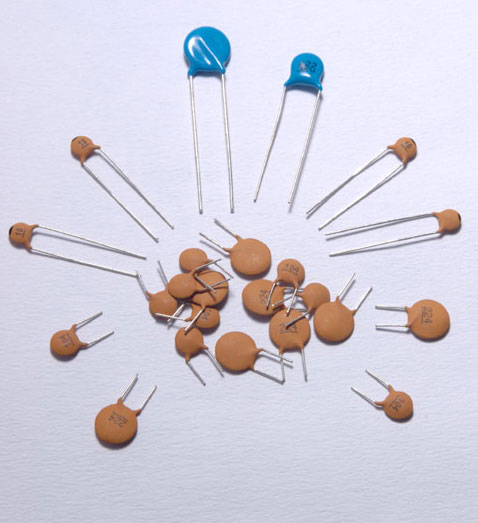
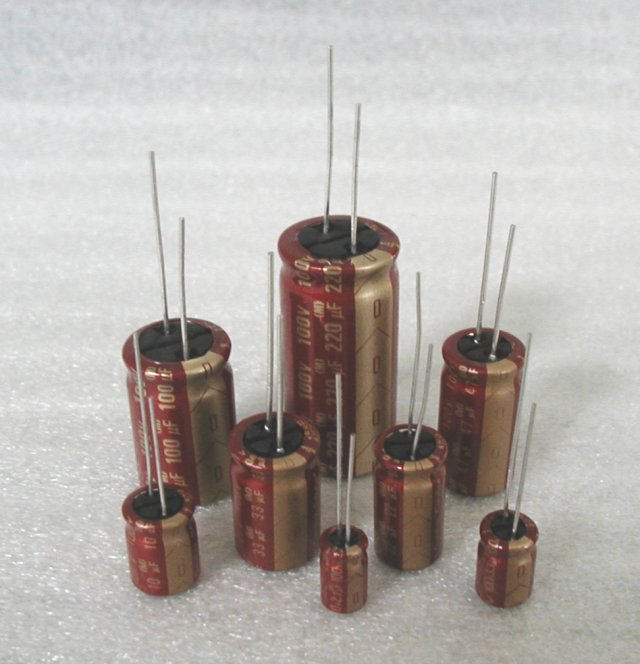
The reciprocal of resistance R is called [conductance](http://en.wikipedia.org/wiki/Conductance) G = 1/R and is measured in [Siemens](http://en.wikipedia.org/wiki/Siemens_%28unit%29) ([SI](http://en.wikipedia.org/wiki/SI) unit), sometimes referred to as a [mho](http://en.wikipedia.org/wiki/Mho). Thus a Siemens is the reciprocal of an ohm: *S* = Ω − 1. Although the concept of conductance is often used in circuit analysis, practical resistors are always specified in terms of their resistance (ohms) rather than conductance.

**CAPACITORS**

A capacitor or condenser is a passive electronic component consisting of a pair of conductors separated by a dielectric. When a voltage potential difference exists between the conductors, an electric field is present in the dielectric. This field stores energy and produces a mechanical force between the plates. The effect is greatest between wide, flat, parallel, narrowly separated conductors.

An ideal capacitor is characterized by a single constant value, capacitance, which is measured in farads. This is the ratio of the electric charge on each conductor to the potential difference between them. In practice, the dielectric between the plates passes a small amount of leakage current. The conductors and leads introduce an equivalent series resistance and the dielectric has an electric field strength limit resulting in a breakdown voltage.

The properties of capacitors in a circuit may determine the resonant frequency and quality factor of a resonant circuit, power dissipation and operating frequency in a digital logic circuit, energy capacity in a high-power system, and many other important aspects.



A capacitor (formerly known as condenser) is a device for storing electric charge. The forms of practical capacitors vary widely, but all contain at least two conductors separated by a non-conductor. Capacitors used as parts of electrical systems, for example, consist of metal foils separated by a layer of insulating film.

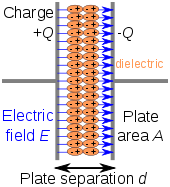
Capacitors are widely used in electronic circuits for blocking [direct current](http://en.wikipedia.org/wiki/Direct_current) while allowing [alternating current](http://en.wikipedia.org/wiki/Alternating_current) to pass, in filter networks, for smoothing the output of [power supplies](http://en.wikipedia.org/wiki/Power_supply), in the [resonant circuits](http://en.wikipedia.org/wiki/LC_circuit) that tune radios to particular [frequencies](http://en.wikipedia.org/wiki/Frequency) and for many other purposes.

A capacitor is a [passive](http://en.wikipedia.org/wiki/Passivity_%28engineering%29)[electronic component](http://en.wikipedia.org/wiki/Electronic_component) consisting of a pair of [conductors](http://en.wikipedia.org/wiki/Electrical_conductor) separated by a [dielectric](http://en.wikipedia.org/wiki/Dielectric) (insulator). When there is a [potential difference](http://en.wikipedia.org/wiki/Potential_difference) (voltage) across the conductors, a static [electric field](http://en.wikipedia.org/wiki/Electric_field) develops in the dielectric that stores [energy](http://en.wikipedia.org/wiki/Energy) and produces a mechanical force between the conductors. An ideal capacitor is characterized by a single constant value, [capacitance](http://en.wikipedia.org/wiki/Capacitance), measured in [farads](http://en.wikipedia.org/wiki/Farad). This is the ratio of the [electric charge](http://en.wikipedia.org/wiki/Electric_charge) on each conductor to the potential difference between them.

The capacitance is greatest when there is a narrow separation between large areas of conductor, hence capacitor conductors are often called "plates", referring to an early means of construction. In practice the dielectric between the plates passes a small amount of [leakage current](http://en.wikipedia.org/wiki/Leakage_%28electronics%29) and also has an electric field strength limit, resulting in a [breakdown voltage](http://en.wikipedia.org/wiki/Breakdown_voltage), while the conductors and [leads](http://en.wikipedia.org/wiki/Lead_%28electronics%29) introduce an undesired [inductance](http://en.wikipedia.org/wiki/Equivalent_series_inductance) and [resistance](http://en.wikipedia.org/wiki/Equivalent_series_resistance).

## Theory of operation

[Capacitance](http://en.wikipedia.org/wiki/Capacitance)

[](http://en.wikipedia.org/wiki/File:Capacitor_schematic_with_dielectric.svg)

[http://bits.wikimedia.org/skins-1.17/common/images/magnify-clip.png](http://en.wikipedia.org/wiki/File:Capacitor_schematic_with_dielectric.svg)

Charge separation in a parallel-plate capacitor causes an internal electric field. A dielectric (orange) reduces the field and increases the capacitance.

[](http://en.wikipedia.org/wiki/File:Plattenkondensator_hg.jpg)

[http://bits.wikimedia.org/skins-1.17/common/images/magnify-clip.png](http://en.wikipedia.org/wiki/File:Plattenkondensator_hg.jpg)

A simple demonstration of a parallel-plate capacitor

A capacitor consists of two [conductors](http://en.wikipedia.org/wiki/Electrical_conductor) separated by a non-conductive region. The non-conductive region is called the dielectric or sometimes the [dielectric medium](http://en.wikipedia.org/wiki/Dielectric_medium). In simpler terms, the dielectric is just an [electrical insulator](http://en.wikipedia.org/wiki/Insulator_%28electrical%29). Examples of dielectric mediums are glass, air, paper, [vacuum](http://en.wikipedia.org/wiki/Vacuum), and even a [semiconductor](http://en.wikipedia.org/wiki/Semiconductor)[depletion region](http://en.wikipedia.org/wiki/Depletion_region) chemically identical to the conductors. A capacitor is assumed to be self-contained and isolated, with no net [electric charge](http://en.wikipedia.org/wiki/Electric_charge) and no influence from any external electric field. The conductors thus hold equal and opposite charges on their facing surfaces, and the dielectric develops an electric field. In [SI](http://en.wikipedia.org/wiki/SI) units, a capacitance of one [farad](http://en.wikipedia.org/wiki/Farad) means that one [coulomb](http://en.wikipedia.org/wiki/Coulomb) of charge on each conductor causes a voltage of one [volt](http://en.wikipedia.org/wiki/Volt) across the device.

The capacitor is a reasonably general model for electric fields within electric circuits. An ideal capacitor is wholly characterized by a constant capacitance C, defined as the ratio of charge ±Q on each conductor to the voltage V between them:

C= \frac{Q}{V}

Sometimes charge build-up affects the capacitor mechanically, causing its capacitance to vary. In this case, capacitance is defined in terms of incremental changes:

C= \frac{\mathrm{d}q}{\mathrm{d}v}

### Energy storage

[Work](http://en.wikipedia.org/wiki/Work_%28thermodynamics%29) must be done by an external influence to "move" charge between the conductors in a capacitor. When the external influence is removed the charge separation persists in the electric field and energy is stored to be released when the charge is allowed to return to its [equilibrium](http://en.wikipedia.org/wiki/Equilibrium) position. The work done in establishing the electric field, and hence the amount of energy stored, is given by:

W= \int_{q=0}^Q V \text{d}q = \int_{q=0}^Q \frac{q}{C} \text{d}q = {1 \over 2} {Q^2 \over C} = {1 \over 2}  C V^2 = {1 \over 2} VQ.

### Current-voltage relation

The current i(t) through any component in an electric circuit is defined as the rate of flow of a charge q(t) passing through it, but actual charges, electrons, cannot pass through the dielectric layer of a capacitor, rather an electron accumulates on the negative plate for each one that leaves the positive plate, resulting in an electron depletion and consequent positive charge on one electrode that is equal and opposite to the accumulated negative charge on the other. Thus the charge on the electrodes is equal to the [integral](http://en.wikipedia.org/wiki/Integral) of the current as well as proportional to the voltage as discussed above. As with any [antiderivative](http://en.wikipedia.org/wiki/Antiderivative), a [constant of integration](http://en.wikipedia.org/wiki/Constant_of_integration) is added to represent the initial voltage v (t0). This is the integral form of the capacitor equation,

v(t)= \frac{q(t)}{C} = \frac{1}{C}\int_{t_0}^t i(\tau) \mathrm{d}\tau+v(t_0).

Taking the derivative of this, and multiplying by C, yields the derivative form,

i(t)= \frac{\mathrm{d}q(t)}{\mathrm{d}t}=C\frac{\mathrm{d}v(t)}{\mathrm{d}t}.

The [dual](http://en.wikipedia.org/wiki/Duality_%28electrical_circuits%29) of the capacitor is the [inductor](http://en.wikipedia.org/wiki/Inductor), which stores energy in the [magnetic field](http://en.wikipedia.org/wiki/Magnetic_field) rather than the electric field. Its current-voltage relation is obtained by exchanging current and voltage in the capacitor equations and replacing C with the inductance L.

**DHT11–TEMPERATURE AND HUMIDITY SENSOR**

**The DHT11**is a commonly used**Temperature and humidity sensor that** comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.



The DHT11 is a commonly used Temperature and humidity sensor. The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers.

The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%. So if you are looking to measure in this range then this sensor might be the right choice for you.

**MAGNETOMETER**

****

The HMC5883L is a triple-axis magnetometer compass, which uses I2C for communication. This Module consists of an on-board voltage regulator allowing you to power the module with voltages between 3.3V and 6V.

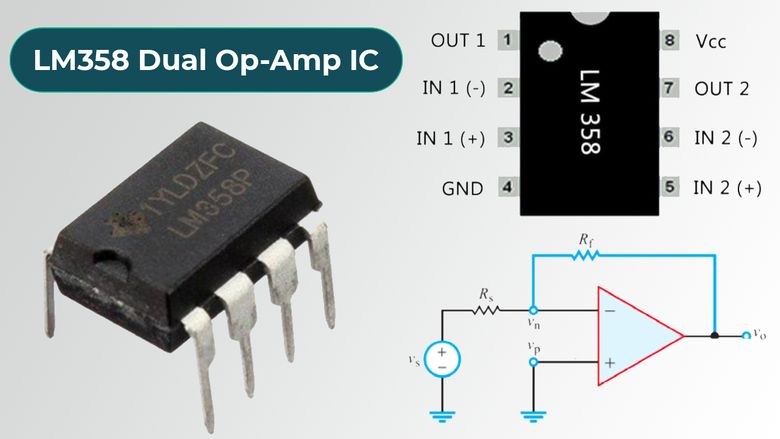
This HMC5883L Magnetometer Module consists of an HMC5883L Magnetometer IC, Voltage Regulator IC, resistors, and capacitors in an integrated circuit. Different manufacturers use a different voltage regulator IC. Most of the modules use XC6206P332MR (662K) IC.

The HMC5883L is a surface-mount, multi-chip module designed for low-field magnetic sensing IC. The HMC5883L includes high-resolution HMC118X series magneto-resistive sensors plus an ASIC containing amplification, automatic degaussing strap drivers, offset cancellation, and a 12-bit ADC that enables 1° to 2° compass heading accuracy.

**Magnetometer Module Features & Specifications**

* Operating Voltage: 3V to 6V DC
* I2C interface
* 1-2 degree heading accuracy
* Integrated 12-bit ADC
* 160Hz max data rate
* Range of -8 to +8 Gauss
* Needs no external components
* Easy to use with Microcontrollers or even with normal Digital/Analog IC
* Small, cheap and easily available

**LM 358**

****

LM358 IC is a dual operational amplifier integrated circuit with two Op-Amp powered by a common power supply. It consists of two independent compensated operational amplifiers with low power and high gain frequency.

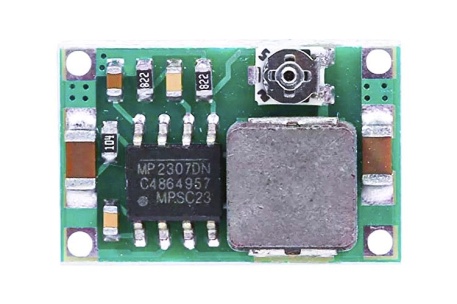
LM358 is specially designed to operate from a single supply over a wide range of voltage. It is more flexible for low voltage AC and moderate voltage DC applications. LM358 is available in a cheap-sized package so it is widely used in real-life applications including transducer amplifier, DC gain block, active filter, and conventional op-amp circuit design. LM358 IC can handle 3V- 32V DC supply and source up to 20 mA per channel.

**LM358 Features and Specification**

Depending upon the manufacturer, each product may have a slightly different variation of the components and each has very similar specifications. Some of these components have identical pinout and package sizes to allow for compatibility between different manufacturers.

* Integrated with two operational amplifiers in a single package
* Wider range of power Supply e.
* 3V to 32V in a single power supply
* ±1.5V to ±16V in a Dual power supply
* The large voltage gain is around 100 dB
* Wider bandwidth in 1 MHz
* Low supply current is 700µA
* The output voltage swing is high
* Differential input voltage range is similar to the power supply voltage
* Short circuit protected outputs
* Internally frequency compensated for unity gain
* Input common-mode voltage range includes ground
* Operating ambient temperature range is 0˚C to 70˚C
* Soldering pin temperature is 260 ˚C
* Available packages are TO-99, CDIP, DSBGA, SOIC, PDIP, DSBGA

**MINI BUCK MODULE**

****

The Mini360 is a small but efficient buck converter module built around the MP2307 from Monolithicpower. This Module is capable of outputting an adjustable DC voltage of 1-17V range, with an input voltage range of 4.75-23V DC. It has a maximum surge current of 3A, while the recommended maximum continuous current is 1.8A. The Output voltage can be adjusted using the onboard potentiometer.

* Features and Specifications
* MPS MP2307DN buck regulator IC
* 4.75 – 23VDC input voltage
* – 17VDC selectable output voltage
* 3A surge current, 1.8A continuous rated current
* Maximum conversion efficiency of 95% (5Vin, 3.3Vout ~200mA output)
* 340kHz switching frequency
* 30mV no-load output ripple
* 0.5% load regulation
* 2.5% voltage regulation
* -40 to +85°C operational temperature range
* Built-in 160°C thermal shutdown
* Dimensions 17x11x3.8mm (0.67×0.43×0.15″)

**SOLAR PANEL**

****

Solar PV panels are comprised of many small photovoltaic cells – photovoltaic meaning they can convert sunlight into electricity. These cells are made of semi-conductive materials, most often silicon, a material that can conduct electricity while maintaining the electrical imbalance needed to create an electric field.

When sunlight hits the semiconductor in the solar PV cell (step 1 in our high level review) the energy from the light, in the form of photons, is absorbed, knocking loose a number of electrons, which then drift freely in the cell. The solar cell is specifically designed with positively and negatively charged semiconductors sandwiched together to create an electric field (see the image to the left for a visualization). This electric field forces the drifting electrons to flow in a certain direction- towards the conductive metal plates that line the cell. This flow is known as an energy current, and the strength of the current determines how much electricity each cell can produce. Once the loose electrons hit metal plates, the current is then directed into wires, allowing the electrons to flow like they would in any other source of electric generation (step 2 in our process).

As the solar panel generates an electric current, the energy flows through a series of wires to an inverter (see step 3 above). While solar panels generate direct current (DC) electricity, most electricity consumers need alternating current (AC) electricity to power their buildings. The inverter’s function is to turn the electricity from DC to AC, making it accessible for everyday use.

After the electricity is transformed into a usable state (AC power), it is sent from the inverter to the electrical panel (also called a breaker box) [step 4], and distributed throughout the building as needed. The electricity is now readily available to power lights, appliances, and other electrical devices with solar energy.

Any electricity that is not consumed via the breaker box is sent to the utility grid through the utility meter (our last step, as outlined above). The utility meter measures the flow of electricity from the grid to your property and vice versa. When your solar energy system is producing more electricity than you are using on site, this meter actually runs backwards, and you are credited for the excess electricity generated through the process of net metering. When you are using more electricity than your solar array is generating, you pull supplemental electricity from the grid through this meter, making it run normally. Unless you have gone completely off-grid through a storage solution, you will need to pull some energy from the grid, especially at night, when your solar array is not producing. However, much of this grid energy will be offset from the excess solar energy you generate throughout the day and in periods of lower usage.

**OLED**

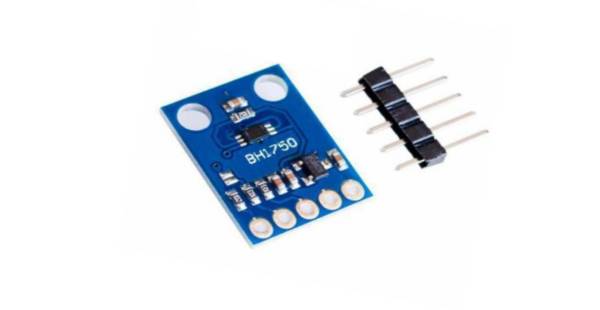
****

To start with let us understand what these **OLED displays** mean. The term OLED stands for “Organic Light emitting diode” it uses the same technology that is used in most of our televisions but has fewer pixels compared to them.  It is real fun to have these cool looking display modules to be interfaced with the Microcontrollers since it will make our projects look cool.

The OLED display doesn’t require backlight, which results in a very nice contrast in dark environments. Additionally, its pixels consume energy only when they are on, so the OLED display consumes less power when compared with other displays.

The model we’re using here has only four pins and communicates with the Arduino using I2C communication protocol. There are models that come with an extra RESET pin. There are also other OLED displays that communicate using SPI communication.

**LIGHT SENSOR**

****

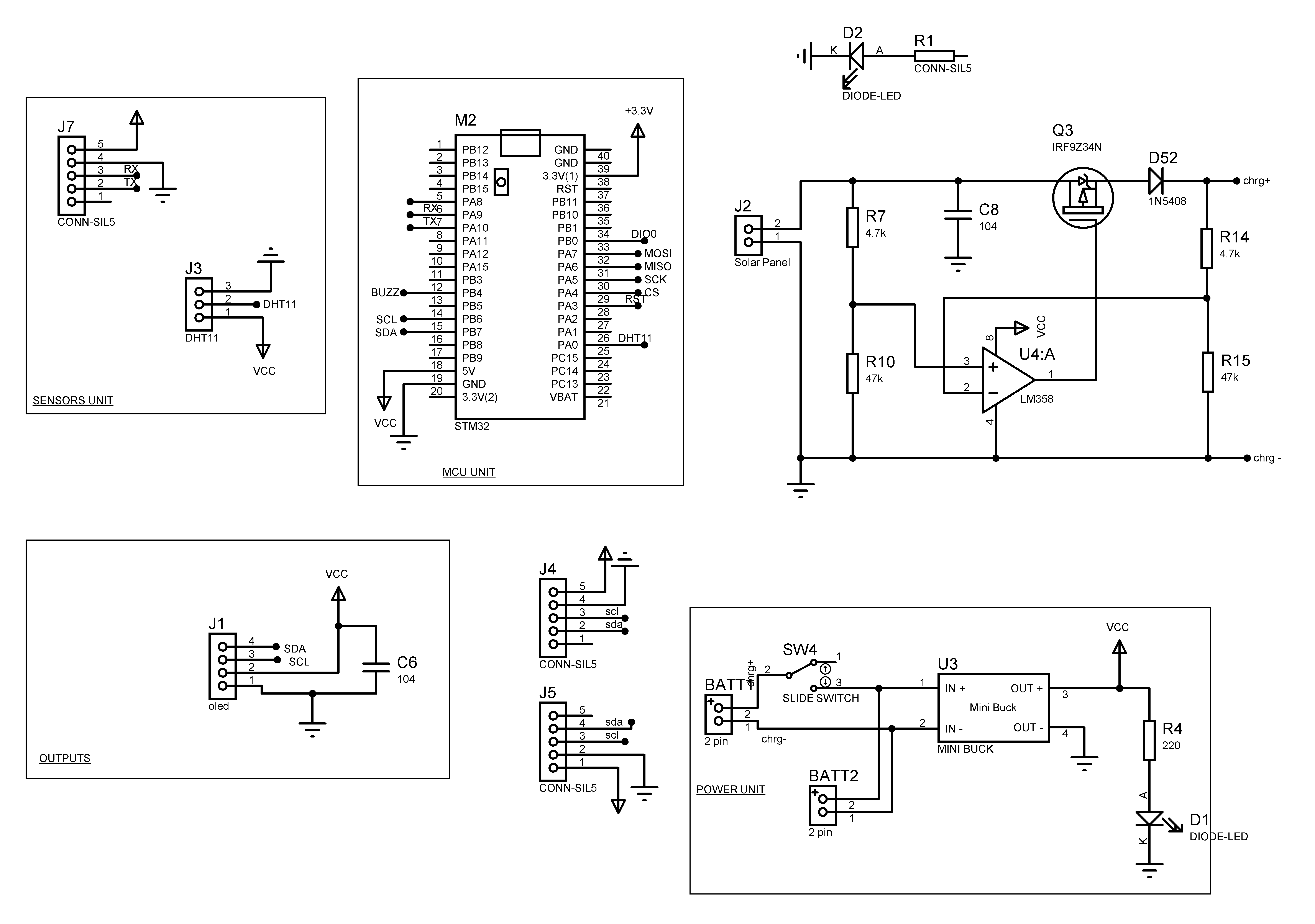
BH1750 is a digital ambient light sensor that is used commonly used in mobile phones to manipulate the screen brightness based on the environment lighting. This sensor can accurately measure the LUX value of light up to 65535lx.

**BH1750 Features**

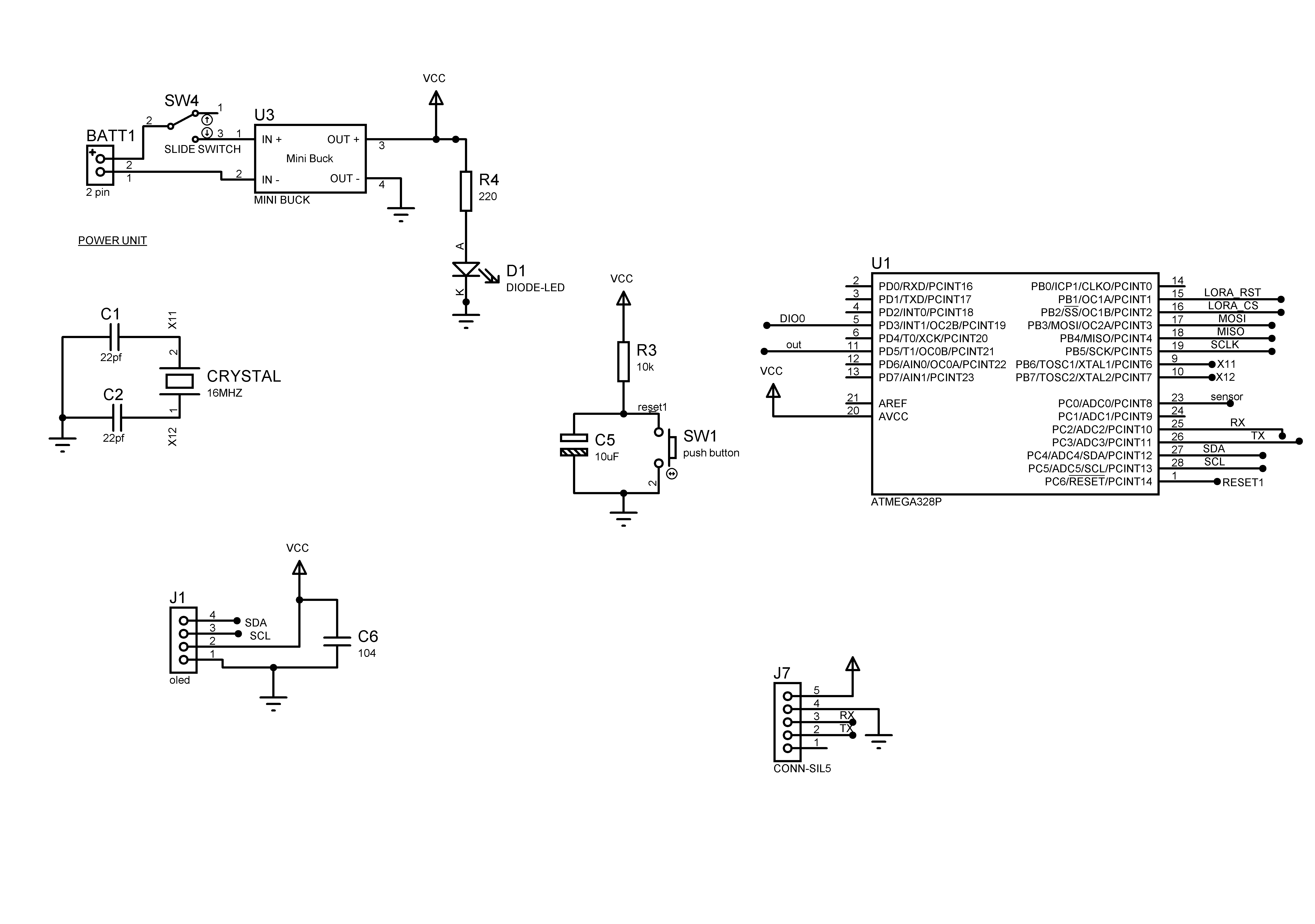
* Power Supply: 2.4V-3.6V (typically 3.0V)
* Less current consumption: 0.12mA
* Measuring Rang: 1-65535lx
* Communication: I2C bus
* Accuracy: +/-20%
* Built in A/D converter for converting analog illuminance in the digital data.
* Very small effect of IR radiation
* Highly responsive near to human eye.

**CIRCUIT DIAGRAM**

**Tx**

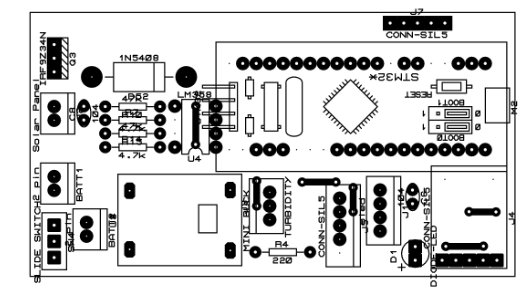
****

**Rx**

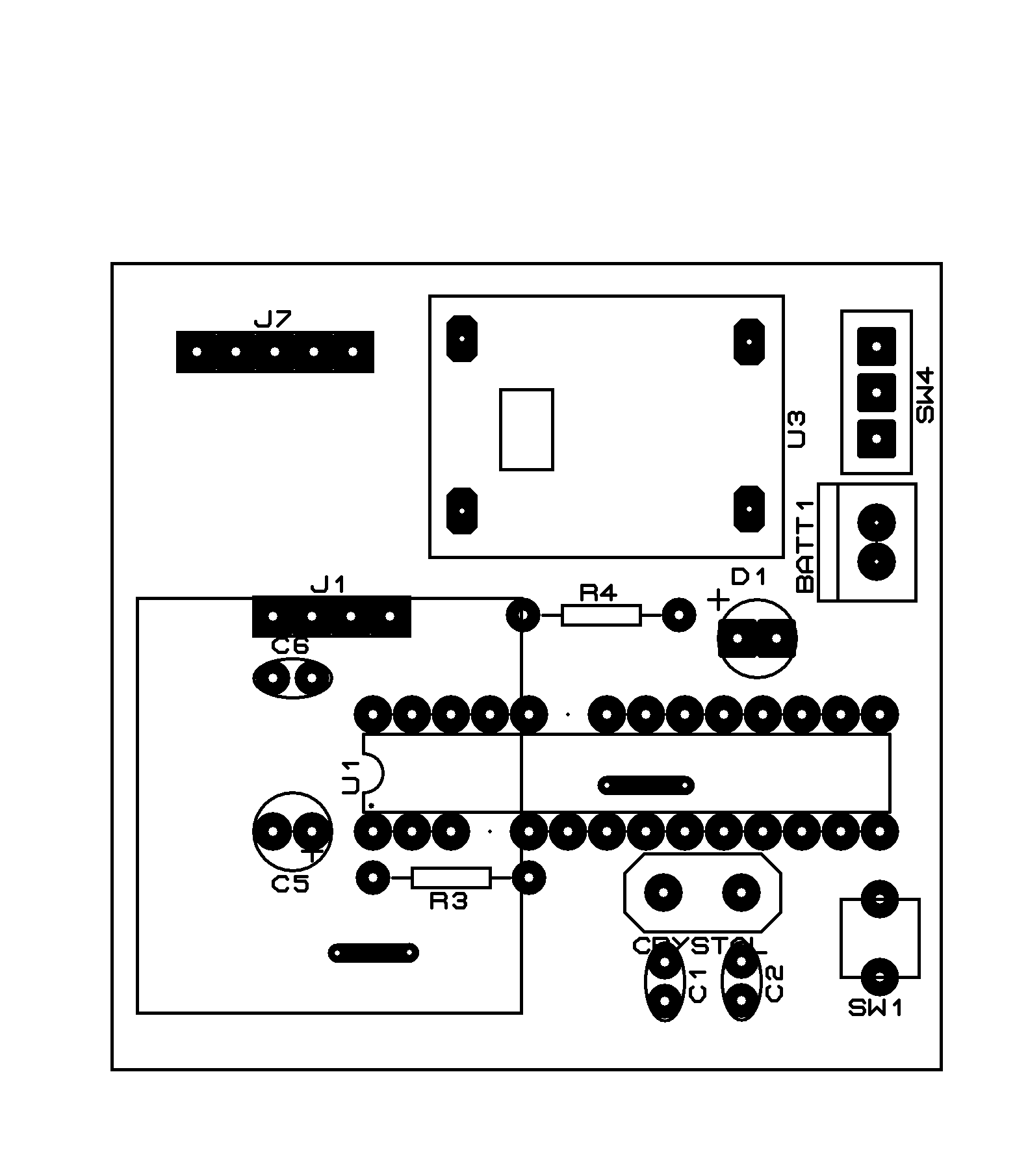
****

**LAYOUT**

**Tx**

****

**Rx**

****

**BOM**

**Tx**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bill Of Materials** | | | |
| **Category** | **Quantity** | **References** | **Value** |
| Modules | 1 | M2 | STM32 |
| Capacitors | 1 | C6 | 104 |
| Capacitors | 1 | C8 | 104 |
| Resistors | 1 | R1 | CONN-SIL5 |
| Resistors | 1 | R4 | 220 |
| Resistors | 2 | R7,R14 | 4.7k |
| Resistors | 2 | R10,R15 | 47k |
| Integrated Circuits | 1 | U3 | MINI BUCK |
| Integrated Circuits | 1 | U4 | LM358 |
| Transistors | 1 | Q3 | IRF9Z34N |
| Diodes | 2 | D1-D2 | DIODE-LED |
| Diodes | 1 | D52 | 1N5408 |
| Miscellaneous | 2 | BATT1-BATT2 | 2 pin |
| Miscellaneous | 1 | J1 | oled |
| Miscellaneous | 1 | J2 | Solar Panel |
| Miscellaneous | 1 | J3 | TURBIDITY |
| Miscellaneous | 3 | J4-J5,J7 | CONN-SIL5 |
| Miscellaneous | 1 | SW4 | SLIDE SWITCH |

**Rx**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bill Of Materials for N548-RX** | | | |
| **Category** | **Quantity** | **References** | **Value** |
| Capacitors | 2 | C1-C2 | 22pf |
| Capacitors | 1 | C5 | 10uF |
| Capacitors | 1 | C6 | 104 |
| Resistors | 1 | R3 | 10k |
| Resistors | 1 | R4 | 220 |
| Integrated Circuits | 1 | U1 | ATMEGA328P |
| Integrated Circuits | 1 | U3 | MINI BUCK |
| Diodes | 1 | D1 | DIODE-LED |
| Miscellaneous | 1 | BATT1 | 2 pin |
| Miscellaneous | 1 | CRYSTAL | 16MHZ |
| Miscellaneous | 1 | J1 | oled |
| Miscellaneous | 1 | J7 | CONN-SIL5 |
| Miscellaneous | 1 | SW1 | push button |
| Miscellaneous | 1 | SW4 | SLIDE SWITCH |

**CODE**

**Tx**

#include <SPI.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306\_STM32.h>

#include <BH1750.h>

#include <QMC5883LCompass.h>

#include "DHT.h"

#define DHTPIN PA0

#define DHTTYPE DHT11

#define OLED\_RESET 4 // Reset pin # (or -1 if sharing Arduino reset pin)

DHT dht(DHTPIN, DHTTYPE);

BH1750 lightMeter;

QMC5883LCompass compass;

Adafruit\_SSD1306 display(OLED\_RESET);

int a;

float lux;

float t;

void setup() {

Serial.begin(115200);//SOFTWARE SERIAL BEGIN

Serial1.begin(9600);

dht.begin();

compass.init();

lightMeter.begin();

display.begin(SSD1306\_SWITCHCAPVCC, 0x3C);

display.clearDisplay();

startScreen();

delay(1000);

display.clearDisplay();

}

void loop() {

t = dht.readTemperature();

compass.read();

a = compass.getAzimuth();

lux = lightMeter.readLightLevel();

delay(500);

MainScreen();

Serial.println(String(t) + "\*" + (String)a + "#" + (String)lux);

Serial1.println(String(t) + "\*" + (String)a + "#" + (String)lux);

}

void startScreen() {

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(20, 2);

display.println(F("Weather Imaging "));

display.setCursor(30, 18);

display.println(F("CubeSat with "));

display.setCursor(40, 33);

display.println(F("Telemetry"));

display.setCursor(30, 50);

display.println(F("Transmission"));

display.display();

delay(1000);

display.clearDisplay();

display.setTextSize(1);

}

void MainScreen()

{

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0, 6);

display.println(F("Orientation:"));

display.setCursor(80, 6);

display.print(a);

display.print((char)247);

display.setCursor(0, 24);

display.println(F("Temperature:"));

display.setCursor(80, 24);

display.print(t);

display.print((char)247);

display.print("C");

display.setCursor(0, 38);

display.println(F("Infrared \nRadiation:"));

display.setCursor(80, 43);

display.print(lux);

display.display();

delay(1000);

display.clearDisplay();

display.setTextSize(1);

}

**Rx**

#include <SoftwareSerial.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

const byte HC12RxdPin = A2; // Recieve Pin on HC12

const byte HC12TxdPin = A3; // Transmit Pin on HC12

String val;

float a;

float t;

float lux;

SoftwareSerial hc12(HC12TxdPin, HC12RxdPin); // Create Software Serial Port

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

#define OLED\_RESET 4 // Reset pin # (or -1 if sharing Arduino reset pin)

#define SCREEN\_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

void setup() {

Serial.begin(115200); // Open serial port to computer

hc12.begin(9600);

if (!display.begin(SSD1306\_SWITCHCAPVCC, SCREEN\_ADDRESS)) {

Serial.println(F("SSD1306 allocation failed"));

for (;;); // Don't proceed, loop forever

}

startScreen();

delay(1000);

display.clearDisplay();

}

void loop()

{

if (hc12.available())

{

val = hc12.readStringUntil('\n');

val.trim();

Serial.print("val = ");

Serial.println(val);

t = val.substring(0, val.indexOf('\*')).toInt();

a = val.substring(val.indexOf('\*') + 1, val.indexOf('#')).toInt();

lux = val.substring(val.indexOf('#') + 1, val.length()).toInt();

Serial.print(t);

Serial.print(a);

Serial.print(lux);

MainScreen();

}

}

void startScreen() {

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(20, 2);

display.println(F("Weather Imaging "));

display.setCursor(30, 18);

display.println(F("CubeSat with "));

display.setCursor(40, 33);

display.println(F("Telemetry"));

display.setCursor(30, 50);

display.println(F("Transmission"));

display.display();

delay(1000);

display.clearDisplay();

display.setTextSize(1);

}

void MainScreen()

{

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0, 6);

display.println(F("Orientation:"));

display.setCursor(80, 6);

display.print(a);

display.print((char)247);

display.setCursor(0, 24);

display.println(F("Temperature:"));

display.setCursor(80, 24);

display.print(t);

display.print((char)247);

display.print("C");

display.setCursor(0, 38);

display.println(F("Infrared \nRadiation:"));

display.setCursor(80, 43);

display.print(lux);

display.display();

delay(1000);

display.clearDisplay();

display.setTextSize(1);

}

void WaitingScreen() {

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(20, 2);

display.println(F("Weather Imaging "));

display.setCursor(30, 18);

display.println(F("00000000 "));

display.setCursor(40, 33);

display.println(F("Telemetry"));

display.setCursor(30, 50);

display.println(F("Transmission"));

display.display();

delay(1000);

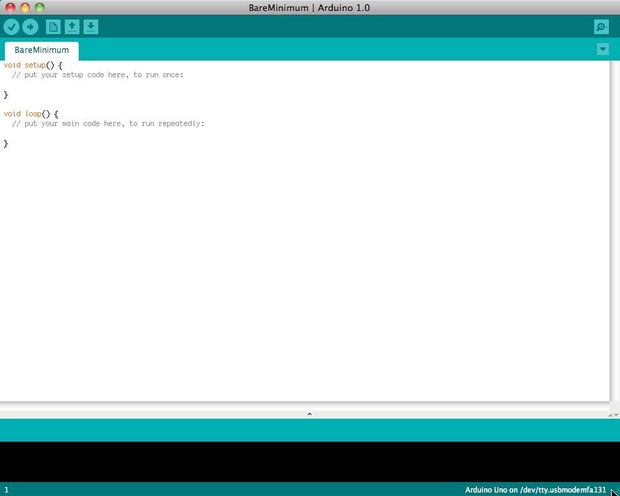
display.clearDisplay();

display.setTextSize(1);

}

**SOFTWARE REQUIREMENTS**

**Arduino IDE**



Before you can start doing anything with the Arduino, you need to download and install the [Arduino IDE](http://www.arduino.cc/en/Main/software) (integrated development environment). From this point on we will be referring to the Arduino IDE as the Arduino Programmer.  
  
The Arduino Programmer is based on the [Processing IDE](http://processing.org/) and uses a variation of the C and C++ programming languages.

## Settings

## Picture of Settings

## cap024.jpg

Before you can start doing anything in the Arduino programmer, you must set the board-type and serial port.  
  
To set the board, go to the following:

Tools --> Boards

Select the version of board that you are using. Since I have an Arduino Uno  plugged in, I obviously selected "Arduino Uno."  
  
To set the serial port, go to the following:

Tools --> Serial Port

Select the serial port that looks like:

/dev/tty.usbmodem [random numbers]

## Picture of Run a sketch

## 6B.jpg

## 6C.jpg

Arduino programs are called sketches. The Arduino programmer comes with a ton of example sketches preloaded. This is great because even if you have never programmed anything in your life, you can load one of these sketches and get the Arduino to do something.  
  
To get the LED tied to digital pin 13 to blink on and off, let's load the blink example.  
  
The blink example can be found here:

Files --> Examples --> Basics --> Blink

## The blink example basically sets pin D13 as an output and then blinks the test LED on the Arduino board on and off every second. Once the blink example is open, it can be installed onto the ATMEGA328 chip by pressing the upload button, which looks like an arrow pointing to the right. Notice that the surface mount status LED connected to pin 13 on the Arduino will start to blink. You can change the rate of the blinking by changing the length of the delay and pressing the upload button again.

## Serial monitor

## Picture of Serial monitor

## cap025.jpg

The serial monitor allows your computer to connect serially with the Arduino. This is important because it takes data that your Arduino is receiving from sensors and other devices and displays it in real-time on your computer. Having this ability is invaluable to debug your code and understand what number values the chip is actually receiving.  
  
For instance, connect center sweep (middle pin) of a potentiometer to A0, and the outer pins, respectively, to 5v and ground. Next upload the sketch shown below:

File --> Examples --> 1.Basics -->AnalogReadSerial

## Click the button to engage the serial monitor which looks like a magnifying glass.  You can now see the numbers being read by the analog pin in the serial monitor. When you turn the knob the numbers will increase and decrease.  The numbers will be between the range of 0 and 1023. The reason for this is that the analog pin is converting a voltage between 0 and 5V to a discreet number.

The Arduino has two different types of input pins, those being analog and digital.  
  
To begin with, lets look at the digital input pins.  
  
Digital input pins only have two possible states, which are on or off. These two on and off states are also referred to as:

* HIGH or LOW
* 1 or 0
* 5V or 0V.

This input is commonly used to sense the presence of voltage when a switch is opened or closed.  
  
Digital inputs can also be used as the basis for countless digital communication protocols. By creating a 5V (HIGH) pulse or 0V (LOW) pulse, you can create a binary signal, the basis of all computing. This is useful for talking to digital sensors like a PING ultrasonic sensor, or communicating with other devices.  
  
For a simple example of a digital input in use, connect a switch from digital pin 2 to 5V, a 10K resistor\*\* from digital pin 2 to ground, and run the following code:

File --> Examples --> 2.Digital --> Button

## Analog in

## Aside from the digital input pins, the Arduino also boasts a number of analog input pins. Analog input pins take an analog signal and perform a 10-bit analog-to-digital (ADC) conversion to turn it into a number between 0 and 1023 (4.9mV steps).  This type of input is good for reading resistive sensors. These are basically sensors which provide resistance to the circuit. They are also good for reading a varying voltage signal between 0 and 5V. This is useful when interfacing with various types of analog circuitry.

## Picture of Write your own code

To write your own code, you will need to learn some basic programming language syntax. In other words, you have to learn how to properly form the code for the programmer to understand it. You can think of this kind of like understanding grammar and punctuation. You can write an entire book without proper grammar and punctuation, but no one will be abler to understand it, even if it is in English.

Some important things to keep in mind when writing your own code:

* An Arduino program is called a sketch.
* All code in an Arduino sketch is processed from top to bottom.
* Arduino sketches are typically broken into five parts.

1. The sketch usually starts with a header that explains what the sketch is doing, and who wrote it.
2. Next, it usually defines global variables. Often, this is where constant names are given to the different Arduino pins.
3. After the initial variables are set, the Arduino begins the setup routine. In the setup function, we set initial conditions of variables when necessary, and run any preliminary code that we only want to run once. This is where serial communication is initiated, which is required for running the serial monitor.
4. From the setup function, we go to the loop routine. This is the main routine of the sketch. This is not only where your main code goes, but it will be executed over and over, so long as the sketch continues to run.
5. Below the loop routine, there is often other functions listed. These functions are user-defined and only activated when called in the setup and loop routine. When these functions are called, the Arduino processes all of the code in the function from top to bottom and then goes back to the next line in the sketch where it left off when the function was called. Functions are good because they allow you to run standard routines - over and over - without having to write the same lines of code over and over. You can simply call upon a function multiple times, and this will free up memory on the chip because the function routine is only written once. It also makes code easier to read. To learn how to form your own functions, check out[this page](http://arduino.cc/en/Reference/FunctionDeclaration).

* All of that said, the only two parts of the sketch which are mandatory are the Setup and Loop routines.
* Code must be written in the [Arduino Language](http://arduino.cc/en/Reference/HomePage), which is roughly based on C.
* Almost all statements written in the Arduino language must end with a ;
* Conditionals (such as [if statements](http://arduino.cc/en/Reference/If) and [for loops](http://arduino.cc/en/Reference/For)) do not need a ;
* Conditionals have their own rules and can be found under "[Control Structures](http://arduino.cc/en/Reference/HomePage)" on the [Arduino Language](http://arduino.cc/en/Reference/HomePage) page
* Variables are storage compartments for numbers. You can pass values into and out of variables. Variables must be defined (stated in the code) before they can be used and need to have a data type associated with it. To learn some of the basic data types, review the [Language Page](http://arduino.cc/en/Reference/HomePage).

Okay! So let us say we want to write code that reads a photocell connected to pin A0, and use the reading we get from the photocell to control the brightness of an LED connected to pin D9.

First, we want to open the BareMinimum sketch, which can be found at:

File --> Examples --> 1.Basic -->BareMinimum

## The BareMinimum Sketch should look like this:

<pre>void setup() {

// put your setup code here, to run once:

}

void loop() {

// put your main code here, to run repeatedly:

}

## Next, lets put a header on the code, so other people know about what we are making, why, and under what terms:

<pre>/\*

LED Dimmer

by Genius Arduino Programmer

2012

Controls the brightness of an LED on pin D9

based on the reading of a photocell on pin A0

This code is in the Public Domain

\*/

void setup() {

// put your setup code here, to run once:

}

void loop() {

// put your main code here, to run repeatedly:

}

## Once that is all squared away, let us define the pin names, and establish variables:

<pre>/\*

LED Dimmer

by Genius Arduino Programmer

2012

Controls the brightness of an LED on pin D9

based on the reading of a photocell on pin A0

This code is in the Public Domain

\*/

// name analog pin 0 a constant name

constintanalogInPin = A0;

// name digital pin 9 a constant name

constintLEDPin = 9;

//variable for reading a photocell

int photocell;

void setup() {

// put your setup code here, to run once:

}

void loop() {

// put your main code here, to run repeatedly:

}

## Now that variables and pin names are set, let us write the actual code:

<pre>/\*

LED Dimmer

by Genius Arduino Programmer

2012

Controls the brightness of an LED on pin D9

based on the reading of a photocell on pin A0

This code is in the Public Domain

\*/

// name analog pin 0 a constant name

constintanalogInPin = A0;

// name digital pin 9 a constant name

constintLEDPin = 9;

//variable for reading a photocell

int photocell;

void setup() {

//nothing here right now

}

void loop() {

//read the analog in pin and set the reading to the photocell variable

photocell = analogRead(analogInPin);

//control the LED pin using the value read by the photocell

analogWrite(LEDPin, photocell);

//pause the code for 1/10 second

//1 second = 1000

delay(100);

}

If we want to see what numbers the analog pin is actually reading from the photocell, we will need to use the serial monitor. Let's activate the serial port and output those numbers:

<pre>/\*

LED Dimmer

by Genius Arduino Programmer

2012

Controls the brightness of an LED on pin D9

based on the reading of a photocell on pin A0

This code is in the Public Domain

\*/

// name analog pin 0 a constant name

constintanalogInPin = A0;

// name digital pin 9 a constant name

constintLEDPin = 9;

//variable for reading a photocell

int photocell;

void setup() {

Serial.begin(9600);

}

void loop() {

//read the analog in pin and set the reading to the photocell variable

photocell = analogRead(analogInPin);

//print the photocell value into the serial monitor

Serial.print("Photocell = " );

Serial.println(photocell);

//control the LED pin using the value read by the photocell

analogWrite(LEDPin, photocell);

//pause the code for 1/10 second

//1 second = 1000

delay(100);

}