**Real time monitoring system for aquaculture using IOT**

**Aim:**

Promote the effective fisheries management and improving standards of fisheries management. Provide the technical and general knowledge necessary for competent fisheries management. Advance the standing of fisheries management as a profession.

**Introduction:**

Aqua culture is one of the flourishing sectors in India as it contributes nearly 1.07% of the GDP. It is estimated that fish requirement of the country by 2025 would be order of 16 million tones but due to over fishing natural fisheries have been depleted as a result commercial aqua culture came into existence. But in recent years commercial aquaculture is facing many problems due to sudden climatic fluctuation which leads to changes in water quality parameters. At present aqua farmers are depending on manual testing for knowing the parameters of water. This will consume time and inaccurate because water quality parameters may alter with time. In order to overcome this problem, technology should be brought to aqua culture which increases the productivity and minimize the losses by constant monitoring of water quality parameters.

**Literature survey:**

1. **“Cyber Aqua Culture Monitoring System Using Arduino And Raspberry Pi,”**

Internet of things is one of the rapidly growing fields for delivering social and economic benefits for emerging and developing economy. The field of IOT is expanding its wings in all the domains like medical, industrial, transportation, education, mining etc. Now-a-days with the advancement in integrated on chip computers like Arduino, Raspberry pi the technology is reaching the ground level with its application in agriculture and aquaculture. Water quality is a critical factor while culturing aquatic organisms. It mainly depends on several parameters like dissolved oxygen, ammonia, pH, temperature, salt, nitrates, carbonates etc. The quality of water is monitored continuously with the help of sensors to ensure growth and survival of aquatic life. The sensed data is transferred to the aqua farmer mobile through cloud. As a result preventive measures can be taken in time to minimize the losses and increase the productivity.

1. “**Water Quality Monitoring and Control for Aquaculture Based on Wireless Sensor Networks”**

We have designed and presented a wireless sensor network monitoring and control system for aquaculture. The system can detect and control water quality parameters of temperature, dissolved oxygen content, pH value, and water level in real-time. The sensor nodes collect the water quality parameters and transmit them to the base station host computer through ZigBee wireless communication standard. The host computer is used for data analysis, processing and presentation using LabVIEW software platform. The water quality parameters will be sent to owners through short messages from the base station via the Global System for Mobile (GSM) module for notification. The experimental evaluation of the network performance metrics of quality of communication link, battery performance and data aggregation was presented. The experimental results show that the system has great prospect and can be used to operate in real world environment for optimum control of aquaculture environment.

**Objectives:**

* Developing the capacity of farmers so that, through their own observations and reasoning, they are able to solve limitations to the implementation of rural aquaculture projects, thereby improving the productivity of household farm units
* Protecting water sources.
* Generating a reliable data base on production and the costs of different production systems.
* Achieving the capacity of farmers to market their agricultural products, including those of aquaculture origin, which contribute to income generation.
* Generating local technological experiences, in agreement with the efficient use of the available resources, and compatible with local cultural traditions and ecological preservation.
* **Block diagram:**

Blynk

Wifi module

Arduino

UNO

PH meter

Mobile

or

PC

Conductivity

Water pump

In-out

Turbidity

Relay

Water level

Solar panel

Power supply system

**Proposed method:**

This proposed work uses an integrated on chip computer Arduino with Wi Fi module which makes it unique. It is energized with the help of solar panel which is more reliable and wireless. Several sensors are mounted to sense the data and the data is transferred to the aqua farmer through IOT. .If the particular values of water quality crosses the threshold range the aqua farmer will get an alert with feasible solution shown in Recently IoT is reaching the ground level with its application to farmers. Several papers in literature survey focuses on how the aquatic life will effect due to change in water quality parameters and how IoT technology is used to overcome the problem.Some papers uses Arduino as micro controller for monitoring the aqua field, Arduino as it has inbuilt Wi-Fi module.Many of the papers concentrates on few type sensors like DO, pH, Turbidity etc and a solution to those problems. All these parameters are sensed using multiple sensors and a feasible solution was given to the aqua farmer.The sensed data will be sent directly to the aqua farmer. But storing the data in cloud database helps us for analyzing the data using data analytics which can help us to take pro-active measures before the change in water quality parameters. The overall system is energized with the help of municipal electricity back up with a battery. But aqua farmers are facing power cuts as a result solar energy can be used as a power module which is more reliable and can move anywhere in the pond without the use of wire. Most of the models concentrates on sending the sensor data to the farmer but our model mainly concentrates on providing the solution such as which medicine should be applied or necessary action to be taken in the form of an alert message when the water quality parameters changes.

**Components required:**

**Hardware requirements:**

* Microcontroller
* Water level Sensor
* Turbidity sensor
* PH sensor
* Temperature sensor
* Conductivity
* Wifi module(ESP8266).
* Android phone
* Power Supply
* Solar panel

**Software requirements:**

* Programming language: Embedded C
* Arduino IDE
* Blynk Android Application

**DESCRIPTION OF COMPONENTS:**

**Temperature sensor****:**

An analog temperature sensor is pretty easy to explain, it's a chip that tells you what the ambient temperature is. These sensors use a solid-state technique to determine the temperature. That is to say, they don't use mercury (like old thermometers), bimetallic strips (like in some home thermometers or stoves), nor do they use thermistors (temperature sensitive resistors). Instead, they use the fact as temperature increases, the voltage across a diode increases at a known rate. (Technically, this is actually the voltage drop between the base and emitter - the Vbe - of a transistor. By precisely amplifying the voltage change, it is easy to generate an analog signal that is directly proportional to temperature. There have been some improvements on the technique but, essentially that is how temperature is measured.

Because these sensors have no moving parts, they are precise, never wear out, don't need calibration, work under many environmental conditions, and are consistent between sensors and readings. Moreover they are very inexpensive and quite easy to use.

**Types of Temperature Sensor:**

There are two temperature sensing methods:

* Contact
* Non-contact

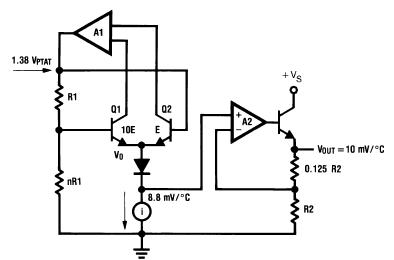
**Contact Temperature Sensor Types** - These types of temperature sensor are required to be inphysical contact with the object being sensed and use conduction to monitor changes in temperature. They can be used to detect solids, liquids or gases over a wide range of temperatures.

**Non-contact Temperature Sensor Types** - These types of temperature sensor use convectionand radiation to monitor changes in temperature. They can be used to detect liquids and gases that emit radiant energy as heat rises and cold settles to the bottom in convection currents or detect the radiant energy being transmitted from an object in the form of infra-red radiation (the sun).

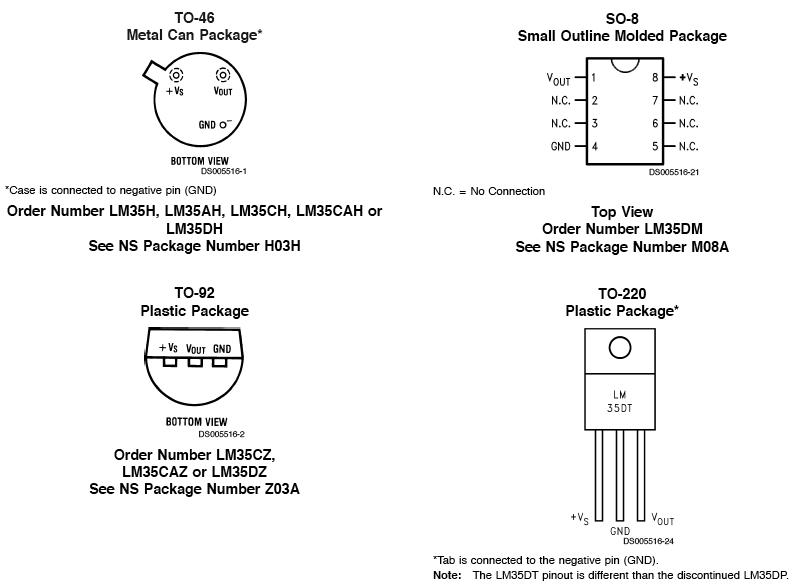
**LM35 Sensor Specification:**

The LM35 series are precision integrated-circuit LM35 temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 sensor thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 sensor does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full -55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a -55° to +150°C temperature range, while the LM35C sensor is rated for a -40° to +110°C range (-10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D sensor is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

**LM35 Sensor Circuit Schematic:**



**LM35 Sensor Pinouts and Packaging:**



**LM35 Sensor Sources:**

There are several manufacturers of this popular part and each has LM35 sensor specs, datasheets and other free LM35 downloads. This amplifier is available from the following manufacturers.

* National Semiconductor
* On Semiconductor
* Texas Instruments
* Fairchild Semiconductor
* STMicroelectronics
* Jameco Electronics
* Analog Devices

**PH meter**

* A **pH Meter** is a [scientific instrument](https://en.wikipedia.org/wiki/Scientific_instrument) that measures the [hydrogen-ion concentration](https://en.wikipedia.org/wiki/Hydrogen-ion_concentration) in water-based solutions, indicating its [acidity](https://en.wikipedia.org/wiki/Acidity) or [alkalinity](https://en.wikipedia.org/wiki/Alkalinity) expressed as [pH](https://en.wikipedia.org/wiki/PH). The pH meter measures the difference in [electrical potential](https://en.wikipedia.org/wiki/Electrical_potential) between a pH electrode and a reference electrode, and so the pH meter is sometimes referred to as a "potentiometric pH meter". The difference in electrical potential relates to the acidity or pH of the solution. The pH meter is used in many applications ranging from [laboratory experimentation](https://en.wikipedia.org/wiki/Experimentation) to [quality control](https://en.wikipedia.org/wiki/Quality_control).
* **Principle of operation**
* [Potentiometric](https://en.wikipedia.org/wiki/Potentiometric) pH meters measure the voltage between two electrodes and display the result converted into the corresponding pH value. They comprise a simple electronic amplifier and a pair of electrodes, or alternatively a combination electrode, and some form of display calibrated in pH units. It usually has a [glass electrode](https://en.wikipedia.org/wiki/Glass_electrode) and a [calomel](https://en.wikipedia.org/wiki/Mercury%28I%29_chloride#Calomel_electrode)[reference electrode](https://en.wikipedia.org/wiki/Reference_electrode), or a combination electrode. The electrodes, or probes, are inserted into the solution to be tested.
* The design of the electrodes is the key part: These are rod-like structures usually made of glass, with a bulb containing the sensor at the bottom. The glass electrode for measuring the pH has a glass bulb specifically designed to be selective to hydrogen-ion concentration. On immersion in the solution to be tested, hydrogen-ions in the test solution exchange for other positively charged ions on the glass bulb, creating an electrochemical potential across the bulb. The electronic amplifier detects the difference in electrical potential between the two electrodes generated in the measurement and converts the potential difference to pH units. The magnitude of the electrochemical potential across the glass bulb is linearly related to the pH according to the [Nernst Equation](https://en.wikipedia.org/wiki/Nernst_Equation).
* The [reference electrode](https://en.wikipedia.org/wiki/Reference_electrode) is insensitive to the pH of the solution being composed of a metallic conductor which connects to the display. This conductor is immersed in an electrolyte solution, typically potassium chloride, which comes into contact with the test solution through a porous ceramic membrane. The display consists of a [voltmeter](https://en.wikipedia.org/wiki/Voltmeter) which displays voltage in units of pH.
* On immersion of the glass electrode and the reference electrode in the test solution, an [electrical circuit](https://en.wikipedia.org/wiki/Electrical_circuit) is completed in which there is a potential difference created and detected by the voltmeter. The circuit can be thought of as going from the conductive element of the reference electrode to the surrounding potassium chloride solution, through the ceramic membrane to the test solution, the hydrogen-ion selective glass of the glass electrode, to the solution inside the glass electrode, to the silver of the glass electrode, and finally the voltmeter of the display device. The voltage varies from test solution to test solution depending on the potential difference created by the difference in hydrogen ion concentrations on each side of the glass membrane between the test solution and the solution inside the glass electrode. All other potential differences in the circuit do not vary with pH and are corrected for by means of the calibration.
* For simplicity, many pH meters use a combination probe, constructed with the glass electrode and the reference electrode contained within a single probe. A detailed description of combination electrodes is given in the article on [glass electrodes](https://en.wikipedia.org/wiki/Glass_electrode).
* The pH meter is [calibrated](https://en.wikipedia.org/wiki/Calibrated) with solutions of known pH, typically before each use, to ensure [accuracy](https://en.wikipedia.org/wiki/Accuracy) of measurement. To measure the pH of a solution, the electrodes are used as probes, which are dipped into the test solutions and held there sufficiently long for the hydrogen-ions in the test solution to [equilibrate](https://en.wikipedia.org/wiki/Dynamic_equilibrium) with the [ions](https://en.wikipedia.org/wiki/Ions) on the surface of the bulb on the glass electrode. This equilibration provides a stable pH measurement.



* **Conductivity**

## Conductivity sensors are compact, fully integrated sensors for measuring the electrical conductivity of seawater. Conductivity is a key parameter for in-situ measurements of several fundamental physical properties of seawater.

* For seawater, the ability to conduct electrical current is mostly dependent on temperature and the amount of inorganic dissolved solids. Salinity is defined as the concentration of dissolved solids. This means that, together with temperature and depth information, a good estimate of the salinity may be determined. By using the inductive principle, stable measurement can be obtained with­out utilizing electrodes that are easily fouled and may wear out in the field.

**Water level sensor**

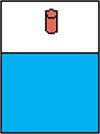
* Wide spectrum of sensors is available in the market and commonly, they are classified based on the specific application of the sensor. Sensor used for measuring humidity is termed as [humidity sensor](http://www.engineersgarage.com/articles/humidity-sensor), the one used for measurement of pressure is called [pressure sensor](http://www.engineersgarage.com/articles/pressure-sensors-types-working), sensor used for measurement of displacement is called [position sensor](http://www.engineersgarage.com/articles/position-sensors) and so on though all of them may be using the similar sensing principle. In a similar fashion, the sensor used for **measurement of fluid levels** is called a level sensor.

* Quite obvious from its name, level sensors are used to measure the level of the free-flowing substances. Such substances include liquids like water, oil, slurries, etc as well as solids in granular/powder form (solids which can flow). These substances tend to get settled in the container tanks due to gravity and maintain their level in rest state. Level sensors measure their level against a pre-set reference.

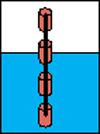
**CLASSIFICATION-BASED ON SENSING POINTS**

* Depending upon the number of location where presence of a fluid (or fluidic solids) is to be sensed, level sensors can be broadly classified under three categories:

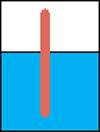
1.      **Single Point Level Sensors** -These sensors are used where fluid level is to be sensed only at single location.



2.      **Multi-point Level Sensors** - These sensors are used where fluid level is to be sensed at number of locations single location.



3.      **Continuous Level Sensors** - These sensors are used where fluid level at all locations is to sensed

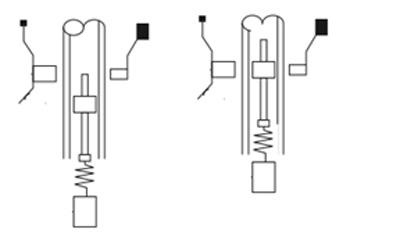


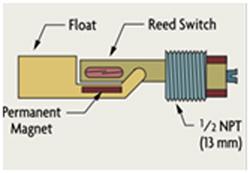
**CLASSFICATION-BASED ON SENSING PRINCIPLES**

A wide variety of sensing principles are used are used for measurement of liquids, fluidic solids, slurries, etc. These are explained below

**·         Float Level Sensors**

In these level sensors, a float moves with the liquid surface. The float is connected to a core via a spring. A magnetic reed switch is mounted in the hermetically sealed core and the core moves inside a stem with the float movement. The stem is encircles by powerful magnets. As the float rises or lowers with liquid level, the reed switch gets operated due to the magnetic field generated by the magnets.



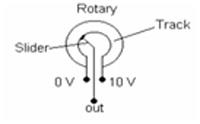


These sensors are also designed by keeping the stem and the core (with magnetic reed switch) stationary and making magnets part of the movable float. For multipoint level sensors multiple magnets/ multiple reed switches (depending upon the design) are used.

The principle of sensors (floats moving with the liquid level) can be coupled to dial gauges. Using buoyancy, they can form Visual liquid level indicators.

**· Resistive Level Sensors**

Variable resistors are widely used in fuel level sensing. A wiper, connected to a lever arm with a float, moves across continuous resistive track.



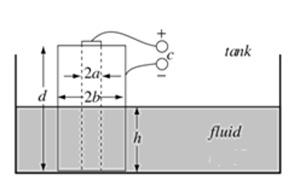
The sensor works on potentiometric measuring principle. Current is made to flow through the resistance. Voltage drops linearly across this resistance. Slider across this resistance is connected to a float. Voltage output is taken between the slider and one end of the resistance. Thus with the varying fluid levels, slider moves and the output voltage varies.

A variant of this type uses conductivity of the liquid under measurement. Current pulses are sent through a sensor electrode (electrically insulated from the tank or external tube). When sensor electrode is immersed into a conductive liquid, an electrical connection is created. The electrical potential is proportional to the liquid level and is measured via a counter-electrode or the tank wall. It is used for continuous filling level measurement and is suitable for all electrically conductive liquids.

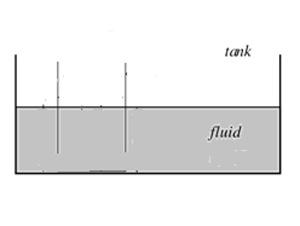
**· Capacitive Level Sensors**

As capacitance depends upon overlapping area between the plates, distance between the plates and the dielectric material between the plates, any of the three can be varies to design a useful capacitive sensor.

One of the simplest **capacitive fluid level sensors** is shown in figure. It comprises of two concentric tubes immersed in the fluid whose level is to be measured. Since the overlapping area between the plates and the distance between the plates is fixed, the capacitance becomes a function of the dielectric between the plates, i.e., fluid between the two concentric tubes. As the fluid level changes, the capacitance also changes. This capacitance becomes the function of the fluid level.



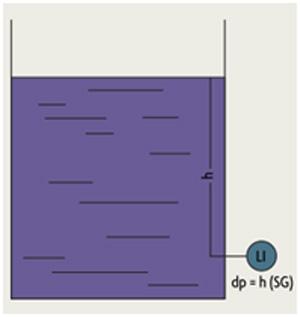
Another variant of this sensor is the one which uses parallel plates instead of concentric tubes. In this case also, change in the fluid level will change the effective dielectric constant and so the capacitance between the plates.



**·     Pressure Based Level Sensors**

Pressure is defined as the force per unit area. The pressure at any depth, in a static fluid is equal to the weight of the liquid acting on a unit area at that depth plus the pressure acting on the surface of the liquid. Level measurement based on pressure measurement is also known as hydrostatic tank gauging.

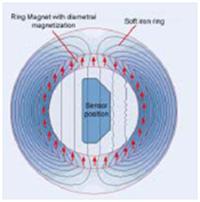
It relies on the principle that the difference between two pressures is equal to the height of the liquid multiplied by specific gravity. So, force at the bottom of the fluid container depends only upon the height of the liquid level and therefore, with the measured hydrostatic pressure and the knowledge of specific gravity of the fluid, level measurement is performed.



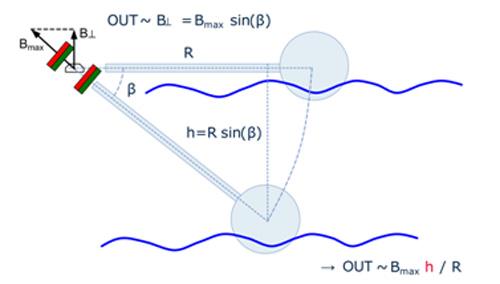
Since these are used for level measurement of corrosive liquids/ water, etc., chemical compatibility of the sensing element should be ascertained. Also, sensors must be calibrated separately for different liquids as the specific gravities are different.

**·         Hall Based Level Sensors**

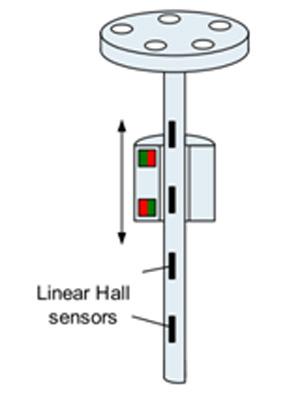
Hall based level sensors have been designed in various configuration. A rotating lever sensor is shown in the figure below.



A linear Hall sensor is placed at the centre of diametrally magnetised ring magnet, surrounded by the soft iron magnet to guide the flux. Hall sensor measures only the vertical component of magnetic field. Thus as the ring moves with the lever, the component of magnetic field measured by the Hall Sensor varies. Thus, the output of Hall sensor becomes a function of the level of the fluid.



Hall Sensors can be used in vertical float systems. Depending on whether continuous or discrete level measurement is needed, an array of hall sensors can be placed at desired points. The magnets become the part of the floats. Hence, with the movement of the float, output of Hall sensors will vary.



Hall based sensors offer good reliability, small dimensions, wide operating voltages and are available at relatively low costs. All these features make them very attractive option amongst variety of other sensors.

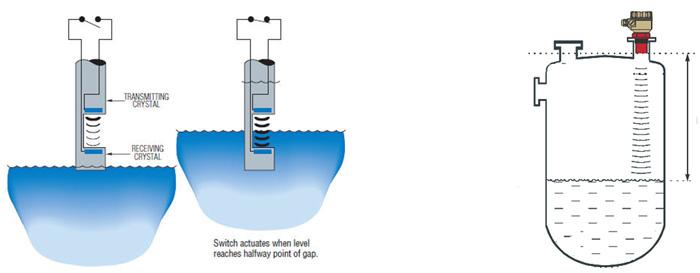
**·         Ultrasonic Level Sensors**

Ultrasonic level instruments operate on the basic time-of-flight principle using sound waves to determine liquid/solid/slurries level.

Ultrasonic Level sensors comprises of two elements; a high efficiency transducer and, an associated electronic transceiver. Complete return trip time between transmitted ultrasonic pulse and reflected echo is measured to determine the fluid level.

The frequency range for ultrasonic methods is in the range of 15...200 kHz. The lower frequency instruments are used for more difficult applications; such as longer distances and solid level measurements and those with higher frequency are used for shorter liquid level measurements.

They can be used as single point **level sensor** or **continuous level sensors**



Also read article on [Ultrasonic Sensors](http://www.engineersgarage.com/articles/ultrasonic-sensors)

**·         Radar Level Sensors**

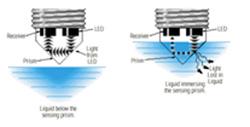
Radar Level Sensors are fundamentally very similar to the ultrasonic levels; only difference between the two is the use of frequencies. Radar level measurement is also based on the principle of measurement of the time elapsed between the transmission of a microwave pulse and the reception of the reflected echo.



Range resolution and frequency are two crucial parameters which should be considered while selecting these sensors. Accuracy of such sensors relies on the application, antenna and its installation, and also on quality of signal processing software.

**·         Optical Sensors**

It is a contact-type sensor and utilizes principle of optical reflection. Theses sensor houses an infrared LED and an IR photodetector. Light emitted from the LED is directed into a prism; the prism forms the tip of the sensor. As long as the prism is out of contact with the liquid, emitted light is reflected back to the receiver. However, when the prism gets immersed in the prism, the light gets refracted out into the liquid, and therefore very little or no light reaches the receiver. Thus, based upon the amount of reflected light, presence or absence of a liquid is sensed.

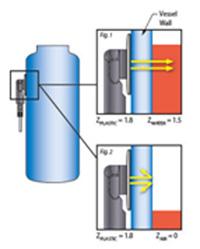


**·**

* **ExoSensors**

ExOsense™ sensor (from Gems Sensors and controls) uses proprietary transducer technology employing piezoelectric material. When piezoelectric material is excited, it creates an acoustic signal as a function of the natural resonance of the material. ExOsense sensors generate this acoustic signal, direct it through the bottle wall and sense the reflection pulse.

The amount of energy that is reflected is determined by the "acoustic impedance mismatch" of the materials in use. For example, if sound passes through two materials with similar acoustic impedances, very little energy will be reflected. If sound passes through two materials with dissimilar impedance values, the majority of the acoustic energy will be reflected. The acoustic impedance mismatch provides the basis for the detection of liquid level.



**SELECTION OF LEVEL** **SENSORS**

There exists extensive variety of commercial solutions available for position sensing and level sensing. Amongst the available options, designers can select best possible technologies to meet their commercial as well as engineering goals. But this also generates a problem of plenty.

Problem of plenty, i.e., availability of too many options often confuse the designers rather than facilitating them. Level sensing, a form a position sensing, can be done using so many different technologies - Inductive, capacitive, mechanical, magneto resistive, Hall effect, optical, etc. The list is not exhaustive. More than one solution may be a viable solution for a particular application and that’s where the confusion arises.

·         Number of questions must be asked while selecting a sensor

·         Points of level sensing required

·         Measurement Range

·         Is the material being measured electrically conductive?

·         Is sensor to be placed inside material or can it be external?

·         Is material solid or liquid?

·         Type of measurement required – contact or non-contact ?

·         Acceptable accuracy, precision and resolution

·         Operating temperature range

·         Type of output – analog, digital, etc.

All these need to be determined for selection the proper sensing technology. Of course, answering these questions is not a straightforward task. But this is where system designer’s skill set is put to test.

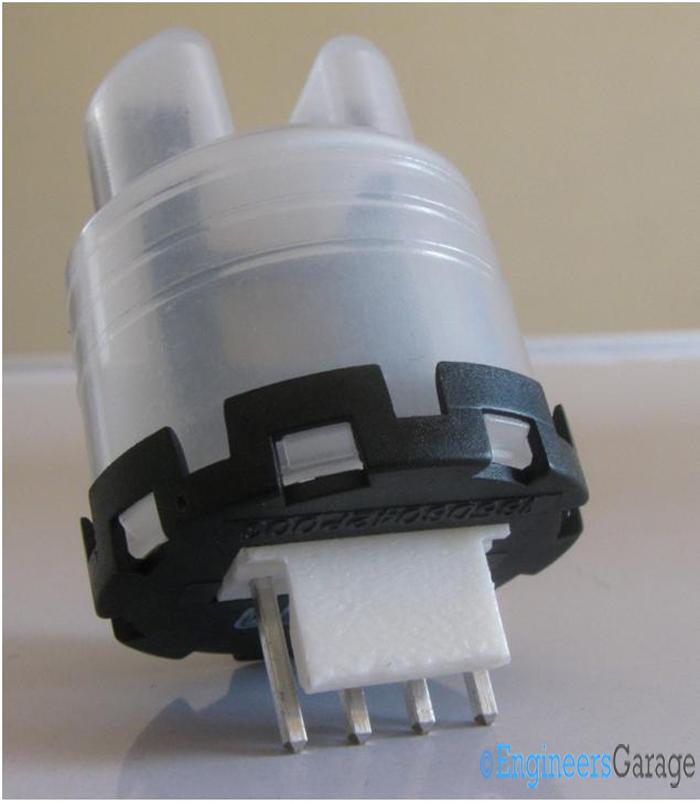
**Turbidity Sensor**

Turbidity is the quantitative measure of suspended particles in a fluid. It can be soil in water or chocolate flakes in your favourite milk shake. While chocolate is something we *so want* in our drinks, soil particles are totally undesired. Keeping aside the potable purposes, there are several industrial and household solutions that make use of water in some or other manner - for instance, a car uses water to clean the windshield, a power plant needs it to cool the reactors, washing machines and dish washers depend on water like fish. Now the question arises here: how do these machines get to know about the turbidity? We are blessed with nature’s evolutionary gift of senses to find out soil in the water, but what about your washing machines? No eyes to see, to tongue to taste, no skin to feel but just a plastic body with some buttons and motor inside. How does it so *smart* to work as per soil suspension?? (If you are wondering does it even do that, the answer is a big yes!)The answer to this is a 4.7 cm gadget: Turbidity Sensor, which along with a micro controller unit, takes care of turbidity measurements. Crafted with plastic and some metal-alloy traces, turbidity sensor uses light to convey information about turbidity in water.

the turbidity sensor appears like an Android bot. Two horn like structure, a top to bottom mono material body. A black colour cap is placed at the bottom of the sensor. Thick alloyed contact legs provide means for various connectors to hold to the sensor. A white plastic slab protects the legs from damage and act as a fixture for good clamping of the sensor.

The plastic used to make outer structure can survive high temperature variations as well as mechanical abrasions.

Scales are found on the transparent can which enable easy gripping. Between the “horns”, a bulge can be seen which holds the thermistor and provides temperature sensing ability to the sensor.



**Arduino Uno**

**Overview**

Arduino is an open-source electronics platform based on easy-to-use hardware and software. [Arduino boards](https://www.arduino.cc/en/Main/Products) are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the [Arduino programming language](https://www.arduino.cc/en/Reference/HomePage) (based on [Wiring](http://wiring.org.co/)), and [the Arduino Software (IDE)](https://www.arduino.cc/en/Main/Software), based on [Processing](https://processing.org/).

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of [accessible knowledge](http://forum.arduino.cc/) that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The [software](https://www.arduino.cc/en/Main/Software), too, is open-source, and it is growing through the contributions of users worldwide.

Arduino is a computer hardware and software company, project, and user community that designs and manufactures [microcontroller](https://en.wikipedia.org/wiki/Microcontroller) kits for building digital devices and interactive objects that can sense and control objects in the physical world. The project's products are distributed as [open-source hardware](https://en.wikipedia.org/wiki/Open-source_hardware) and [software](https://en.wikipedia.org/wiki/Open-source_software), which are licensed under the [GNU Lesser General Public License](https://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License) (LGPL) or the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License) (GPL),[[1]](https://en.wikipedia.org/wiki/Arduino#cite_note-1) permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form, or as [do-it-yourself](https://en.wikipedia.org/wiki/Do-it-yourself) kits.

The project's board designs use a variety of microprocessors and controllers. These systems provide sets of digital and analog [input/output](https://en.wikipedia.org/wiki/Input/output) (I/O) pins that may be interfaced to various expansion boards ("shields") and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus ([USB](https://en.wikipedia.org/wiki/USB)) on some models, for loading programs from personal computers. The microcontrollers are mainly programmed using a dialect of features from the programming languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B). In addition to using traditional compiler toolchains, the Arduino project provides an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) based on the [Processing](https://en.wikipedia.org/wiki/Processing_(programming_language)) language project.

The Arduino project started in 2005 as a program for students at the [Interaction Design Institute Ivrea](https://en.wikipedia.org/wiki/Interaction_Design_Institute_Ivrea) in [Ivrea](https://en.wikipedia.org/wiki/Ivrea" \o "Ivrea), Italy,[[2]](https://en.wikipedia.org/wiki/Arduino#cite_note-kushner-2) aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using [sensors](https://en.wikipedia.org/wiki/Sensor) and [actuators](https://en.wikipedia.org/wiki/Actuator). Common examples of such devices intended for beginner hobbyists include simple [robots](https://en.wikipedia.org/wiki/Robot), [thermostats](https://en.wikipedia.org/wiki/Thermostat), and [motion detectors](https://en.wikipedia.org/wiki/Motion_detector).

Arduino/Genuino Uno is a microcontroller board based on the ATmega328P ([datasheet](http://www.atmel.com/Images/doc8161.pdf)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header 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 a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

### Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

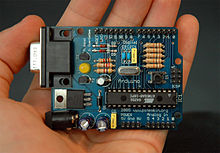
There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

* **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
* **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
* **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
* **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
* **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the [breadboard version of the module](https://www.arduino.cc/en/Main/Standalone) in order to understand how it works and save money.

**Hardware**

Arduino is [open-source hardware](https://en.wikipedia.org/wiki/Open-source_hardware). The hardware reference designs are distributed under a [Creative Commons](https://en.wikipedia.org/wiki/Creative_Commons) Attribution Share-Alike 2.5 license and are available on the Arduino website. Layout and production files for some versions of the hardware are also available. The source code for the IDE is released under the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License), version 2.[[8]](https://en.wikipedia.org/wiki/Arduino#cite_note-8) Nevertheless an official [Bill of Materials](https://en.wikipedia.org/wiki/Bill_of_Materials) of Arduino boards has never been released by the staff of Arduino.

Although the hardware and software designs are freely available under [copyleft](https://en.wikipedia.org/wiki/Copyleft" \o "Copyleft) licenses, the developers have requested that the name "Arduino" be [exclusive to the official product](https://en.wikipedia.org/wiki/Generic_trademark) and not be used for derived works without permission. The official policy document on use of the Arduino name emphasizes that the project is open to incorporating work by others into the official product.[[9]](https://en.wikipedia.org/wiki/Arduino#cite_note-AutoF7-44-9) Several Arduino-compatible products commercially released have avoided the *Arduino* name by using *-duino* name variants.[[10]](https://en.wikipedia.org/wiki/Arduino#cite_note-freeduino-10)

[](https://en.wikipedia.org/wiki/File:Arduino316.jpg)

An early Arduino board[[11]](https://en.wikipedia.org/wiki/Arduino#cite_note-11) with an [RS-232](https://en.wikipedia.org/wiki/RS-232) [serial](https://en.wikipedia.org/wiki/Serial_communication) interface (upper left) and an Atmel ATmega8 microcontroller chip (black, lower right); the 14 digital I/O pins are at the top, the 6 analog input pins at the lower right, and the power connector at the lower left.

An Arduino board consists of an [Atmel](https://en.wikipedia.org/wiki/Atmel) 8-, 16- or 32-bit AVR [microcontroller](https://en.wikipedia.org/wiki/Microcontroller) (although since 2015 other makers' microcontrollers have been used) with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which let users connect the CPU board to a variety of interchangeable add-on modules termed *shields*. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an [I²C](https://en.wikipedia.org/wiki/I%C2%B2C)[serial bus](https://en.wikipedia.org/wiki/Serial_bus)—so many shields can be stacked and used in parallel. Before 2015, Official Arduinos had used the Atmel [megaAVR](https://en.wikipedia.org/wiki/MegaAVR" \o "MegaAVR) series of chips, specifically the ATmega8, ATmega168, [ATmega328](https://en.wikipedia.org/wiki/ATmega328), ATmega1280, and ATmega2560. In 2015, units by other producers were added. A handful of other processors have also been used by Arduino compatible devices. Most boards include a 5 V [linear regulator](https://en.wikipedia.org/wiki/Linear_regulator) and a 16 MHz [crystal oscillator](https://en.wikipedia.org/wiki/Crystal_oscillator) (or [ceramic resonator](https://en.wikipedia.org/wiki/Ceramic_resonator) in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a [boot loader](https://en.wikipedia.org/wiki/Boot_loader) that simplifies uploading of programs to the on-chip [flash memory](https://en.wikipedia.org/wiki/Flash_memory), compared with other devices that typically need an external [chip programmer](https://en.wikipedia.org/wiki/Programmer_(hardware)). This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer. Currently, optibootbootloader is the default bootloader installed on Arduino UNO.[[12]](https://en.wikipedia.org/wiki/Arduino#cite_note-12)

At a conceptual level, when using the Arduino integrated development environment, all boards are programmed over a serial connection. Its implementation varies with the hardware version. Some serial Arduino boards contain a level shifter circuit to convert between [RS-232](https://en.wikipedia.org/wiki/RS-232) logic levels and [transistor–transistor logic](https://en.wikipedia.org/wiki/Transistor%E2%80%93transistor_logic) (TTL) level signals. Current Arduino boards are programmed via [Universal Serial Bus](https://en.wikipedia.org/wiki/Universal_Serial_Bus) (USB), implemented using USB-to-serial adapter chips such as the [FTDI](https://en.wikipedia.org/wiki/FTDI) FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, [Bluetooth](https://en.wikipedia.org/wiki/Bluetooth) or other methods, when used with traditional microcontroller tools instead of the Arduino IDE, standard AVR [in-system programming](https://en.wikipedia.org/wiki/In-system_programming) (ISP) programming is used.

[](https://en.wikipedia.org/wiki/File:UnoConnections.jpg)

An official Arduino Uno R2 with descriptions of the I/O locations

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The *Diecimila*,[[a]](https://en.wikipedia.org/wiki/Arduino#cite_note-N10000-13) *Duemilanove*,[[b]](https://en.wikipedia.org/wiki/Arduino#cite_note-N2009-14) and current *Uno*[[c]](https://en.wikipedia.org/wiki/Arduino#cite_note-N1-15) provide 14 digital I/O pins, six of which can produce [pulse-width modulated](https://en.wikipedia.org/wiki/Pulse-width_modulation) signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board[[13]](https://en.wikipedia.org/wiki/Arduino" \l "cite_note-16) and Boarduino[[14]](https://en.wikipedia.org/wiki/Arduino" \l "cite_note-17) boards may provide male header pins on the underside of the board that can plug into solderless [breadboards](https://en.wikipedia.org/wiki/Breadboard).

Many Arduino-compatible and Arduino-derived boards exist. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education, to simplify making buggies and small robots. Others are electrically equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not. Some variants use different processors, of varying compatibility.

## Digital Pins

The pins on the Arduino can be configured as either inputs or outputs. This document explains the functioning of the pins in those modes. While the title of this document refers to digital pins, it is important to note that vast majority of Arduino (Atmega) analog pins, may be configured, and used, in exactly the same manner as digital pins.

### Properties of Pins Configured as INPUT

Arduino (Atmega) pins default to inputs, so they don't need to be explicitly declared as inputs with pinMode() when you're using them as inputs. Pins configured this way are said to be in a **high-impedance state**. Input pins make extremely small demands on the circuit that they are sampling, equivalent to a series resistor of 100 megohm in front of the pin. This means that it takes very little current to move the input pin from one state to another, and can make the pins useful for such tasks as implementing [a capacitive touch sensor](http://www.arduino.cc/playground/Code/CapacitiveSensor), reading an LED as a [photodiode](http://www.arduino.cc/playground/Learning/LEDSensor), or reading an analog sensor with a scheme such as [RCTime.](https://www.arduino.cc/en/Tutorial/RCtime)

This also means however, that pins configured as pinMode(pin, INPUT) with nothing connected to them, or with wires connected to them that are not connected to other circuits, will report seemingly random changes in pin state, picking up electrical noise from the environment, or capacitively coupling the state of a nearby pin.

### Pullup Resistors with pins configured as INPUT

Often it is useful to steer an input pin to a known state if no input is present. This can be done by adding a pullup resistor (to +5V), or a pulldown resistor (resistor to ground) on the input. A 10K resistor is a good value for a pullup or pulldown resistor.

### Properties of Pins Configured as INPUT\_PULLUP

There are 20K pullup resistors built into the Atmega chip that can be accessed from software. These built-in pullup resistors are accessed by setting the pinMode() as INPUT\_PULLUP. This effectively inverts the behavior of the INPUT mode, where HIGH means the sensor is off, and LOW means the sensor is on.

The value of this pullup depends on the microcontroller used. On most AVR-based boards, the value is guaranteed to be between 20kΩ and 50kΩ. On the Arduino Due, it is between 50kΩ and 150kΩ. For the exact value, consult the datasheet of the microcontroller on your board.

When connecting a sensor to a pin configured with INPUT\_PULLUP, the other end should be connected to ground. In the case of a simple switch, this causes the pin to read HIGH when the switch is open, and LOW when the switch is pressed.

The pullup resistors provide enough current to dimly light an LED connected to a pin that has been configured as an input. If LEDs in a project seem to be working, but very dimly, this is likely what is going on.

The pullup resistors are controlled by the same registers (internal chip memory locations) that control whether a pin is HIGH or LOW. Consequently, a pin that is configured to have pullup resistors turned on when the pin is an INPUT, will have the pin configured as HIGH if the pin is then switched to an OUTPUT with pinMode(). This works in the other direction as well, and an output pin that is left in a HIGH state will have the pullup resistors set if switched to an input with pinMode().

### Properties of Pins Configured as OUTPUT

Pins configured as OUTPUT with pinMode() are said to be in a low-impedance state. This means that they can provide a substantial amount of current to other circuits. Atmega pins can source (provide positive current) or sink (provide negative current) up to 40 mA (milliamps) of current to other devices/circuits. This is enough current to brightly light up an LED (don't forget the series resistor), or run many sensors, for example, but not enough current to run most relays, solenoids, or motors.

Short circuits on Arduino pins, or attempting to run high current devices from them, can damage or destroy the output transistors in the pin, or damage the entire Atmega chip. Often this will result in a "dead" pin in the microcontroller but the remaining chip will still function adequately. For this reason it is a good idea to connect OUTPUT pins to other devices with 470Ω or 1k resistors, unless maximum current draw from the pins is required for a particular

Technical specifications

|  |  |
| --- | --- |
| Microcontroller | [ATmega328P](http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf) |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (ATmega328P) of which 0.5 KB used by bootloader |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Clock Speed | 16 MHz |
| LED\_BUILTIN | 13 |
| Length | 68.6 mm |
| Width | 53.4 mm |
| Weight | 25 g |
|  |  |

**ATmega328p Microcontroller**

# Introduction

The Atmel®picoPower® ATmega328/P is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega328/P achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed.FreaturesHigh Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family 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 20MHz
* On-chip 2-cycle Multiplier

High Endurance Non-volatile Memory Segments

* 32KBytes of In-System Self-Programmable Flash program

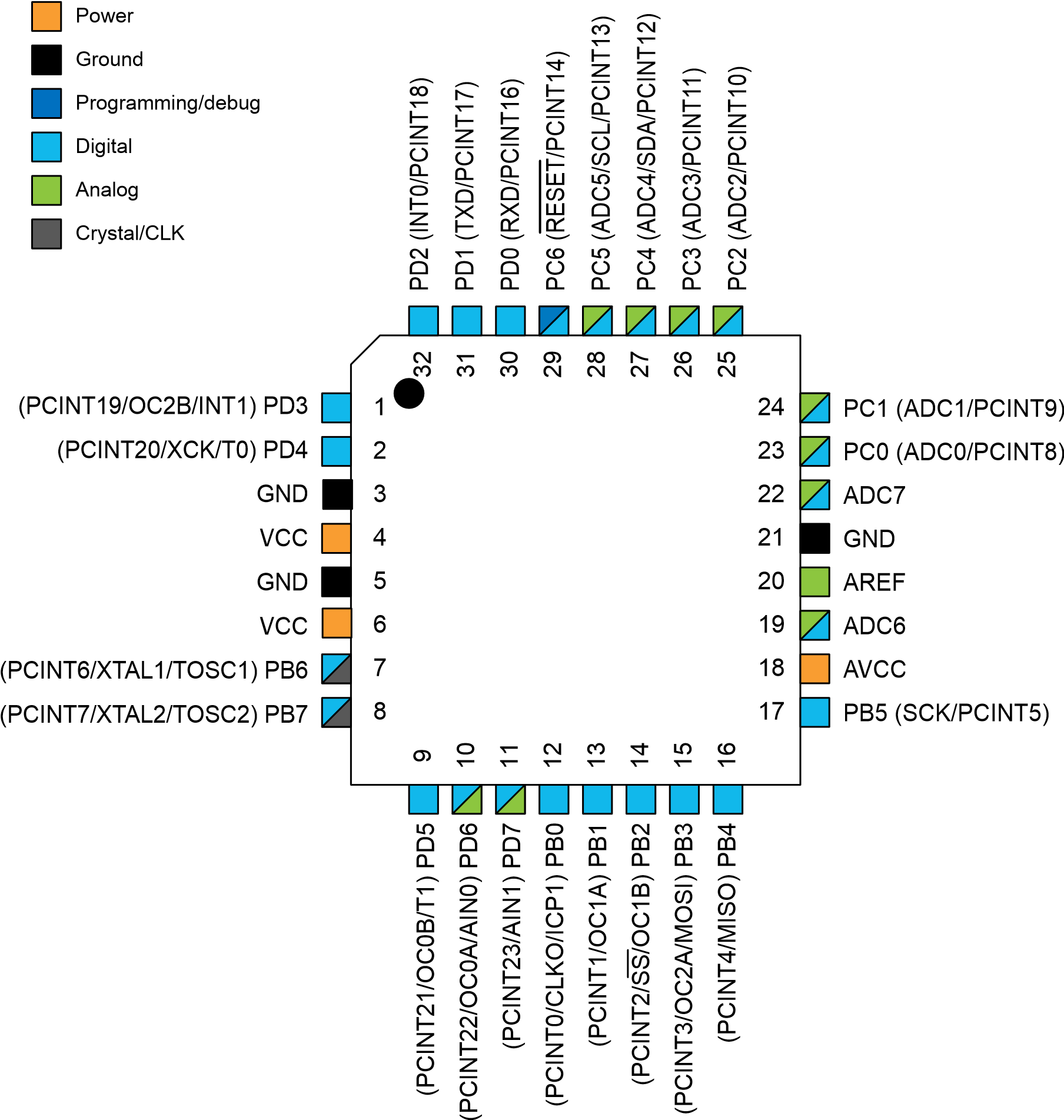
**Memory**

* 1KBytes EEPROM
* 2KBytes Internal SRAM
* 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
* Atmel®QTouch® Library Support
* Capacitive Touch Buttons, Sliders and Wheels
* QTouch and QMatrix® Acquisition
* Up to 64 sense channels
* 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
* Six PWM Channels
* 8-channel 10-bit ADC in TQFP and QFN/MLF package
* Temperature Measurement
* 6-channel 10-bit ADC in PDIP Package
* Two Master/Slave SPI Serial Interface
* One Programmable Serial USART
* One Byte-oriented 2-wire Serial Interface (Philips I2C compatible)
* Programmable Watchdog Timer with Separate On-chip Oscillator
* One 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
* Temperature Range:
* -40°C to 105°C
* Speed Grade:
* 0 - 4MHz @ 1.8 - 5.5V
* 0 - 10MHz @ 2.7 - 5.5V
* 0 - 20MHz @ 4.5 - 5.5V
* Power Consumption at 1MHz, 1.8V, 25°C
* Active Mode: 0.2mA
* Power-down Mode: 0.1μA
* Power-save Mode: 0.75μA (Including 32kHz RTC)

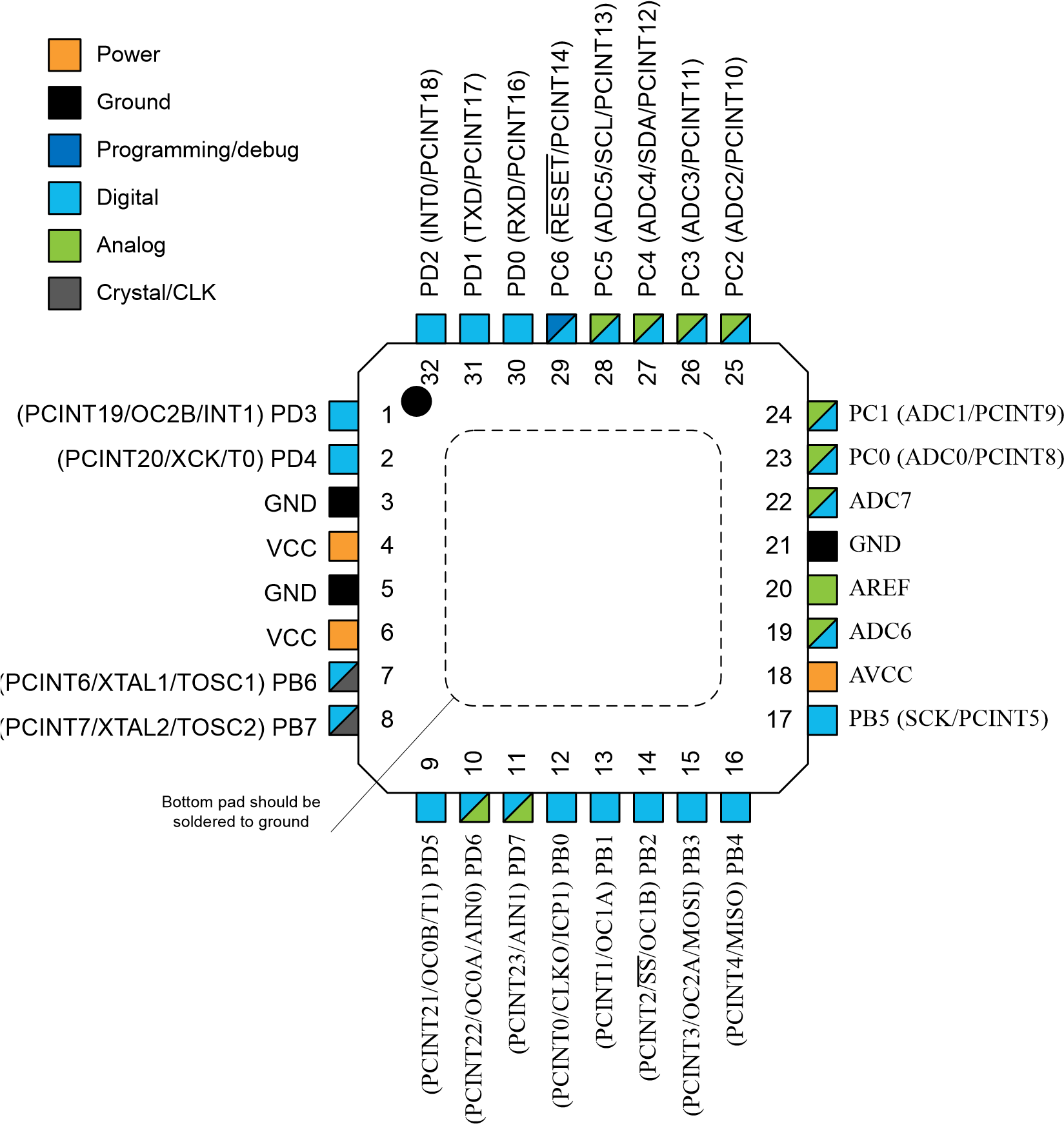
## Pin Configurations

28-pin MLF Top View

#### Figure 5-3. 32-pin TQFP Top View



#### Figure 5-4. 32-pin MLF Top View



### Pin Descriptions

### VCC

Digital supply voltage.

**GND**

Ground.

**Port B (PB[7:0]) XTAL1/XTAL2/TOSC1/TOSC2**

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

If the Internal Calibrated RC Oscillator is used as chip clock source, PB[7:6] is used as TOSC[2:1] input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

**Port C (PC[5:0])**

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC[5:0] output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

**PC6/RESET**

If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C.

If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a Reset.

The various special features of Port C are elaborated in the *Alternate Functions of Port C* section.

**Port D (PD[7:0])**

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

**AVCC**

AVCC is the supply voltage pin for the A/D Converter, PC[3:0], and PE[3:2]. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC[6:4] use digital supply voltage, VCC.

**AREF**

AREF is the analog reference pin for the A/D Converter.

**ADC[7:6] (TQFP and VFQFN Package Only)**

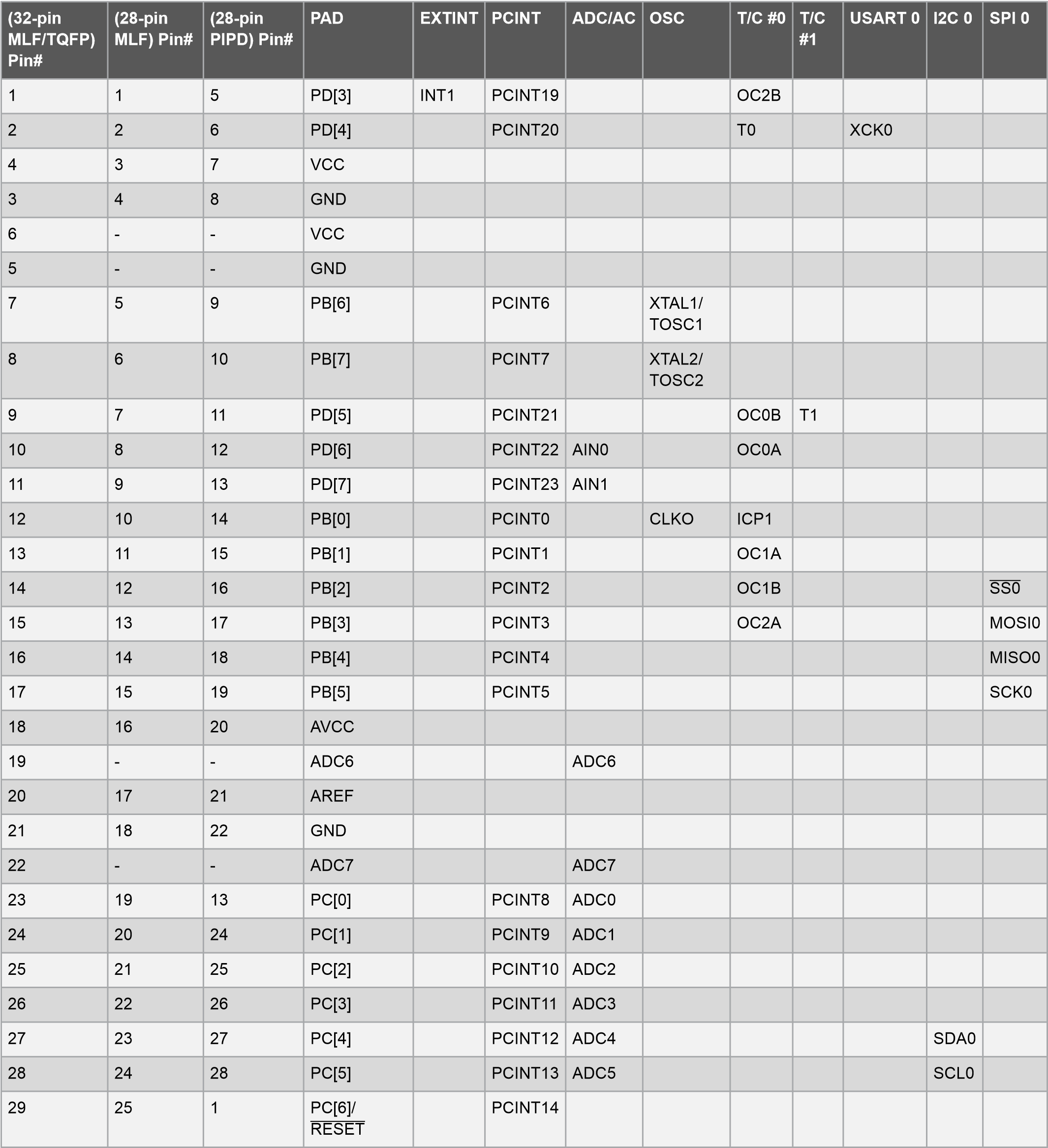
In the TQFP and VFQFN package, ADC[7:6] serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

## 6.I/O Multiplexing

Each pin is by default controlled by the PORT as a general purpose I/O and alternatively it can be assigned to one of the peripheral functions.

The following table describes the peripheral signals multiplexed to the PORT I/O pins.

### Table 6-1. PORT Function Multiplexing



|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **(32-pin**  **MLF/TQFP)**  **Pin#** | **(28-pin MLF) Pin#** | **(28-pin**  **PIPD) Pin#** | **PAD** | **EXTINT** | **PCINT** | **ADC/AC** | **OSC** | **T/C #0** | **T/C**  **#1** | **USART 0** | **I2C 0** | **SPI 0** |
| 30 | 26 | 2 | PD[0] |  | PCINT16 |  |  |  |  | RXD0 |  |  |
| 31 | 27 | 3 | PD[1] |  | PCINT17 |  |  |  |  | TXD0 |  |  |
| 32 | 28 | 4 | PD[2] | INT0 | PCINT18 |  |  |  |  |  |  |  |

**ESP8266 (WIFI MODULE):**

The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Arduino device and get about as much WiFi-ability as a WiFi Shield offers (and that’s just out of the box)! The ESP8266 module is an extremely cost effective board with a huge, and ever growing, community.

This module has a powerful enough on-board processing and storage capability that allows it sto be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, is designed to occupy minimal PCB area. The ESP8266 supports APSD for VoIP applications and Bluetooth co-existanceinterfaces, it contains a self-calibrated RF allowing it to work under all operating conditions, and requires no external RF parts.

There is an almost limitless fountain of information available for the ESP8266, all of which has been provided by amazing community support. In the *Documents* section below you will find many resources to aid you in using the ESP8266, even instructions on how to transforming this module into an IoT (Internet of Things) solution!

**Note:** The ESP8266 Module is not capable of 5-3V logic shifting and will require an external [Logic Level Converter](https://www.sparkfun.com/products/12009). Please do not power it directly from your 5V dev board.

**Note:** This new version of the ESP8266 WiFi Module has increased the flash disk size from 512k to 1MB.

**Features:**

* 802.11 b/g/n
* Wi-Fi Direct (P2P), soft-AP
* Integrated TCP/IP protocol stack
* Integrated TR switch, balun, LNA, power amplifier and matching network
* Integrated PLLs, regulators, DCXO and power management units
* +19.5dBm output power in 802.11b mode
* Power down leakage current of <10uA
* 1MB Flash Memory
* Integrated low power 32-bit CPU could be used as application processor
* SDIO 1.1 / 2.0, SPI, UART
* STBC, 1×1 MIMO, 2×1 MIMO
* A-MPDU & A-MSDU aggregation & 0.4ms guard interval
* Wake up and transmit packets in < 2ms
* Standby power consumption of < 1.0mW (DTIM3)

**Wi-Fi Module:**

* Wireless Technology is an alternative to Wired Technology which is commonly used for connecting devices in wireless mode.
* Wi-Fi (Wireless Fidelity) is a generic term that refers to IEEE 802.11 standard for Wireless Local Area Networks (WLANs).
* Wi-Fi Network connect computers to each other, to the internet and to the wired network.

Wi-Fi Networks use Radio Technologies to transmit and receive data at high speed:

* + IEEE 802.11b
  + IEEE 802.11a
  + IEEE 802.11g

**IEEE 802.11b:**

* Appear in late 1999
* Operates at 2.4GHz radio spectrum
* 11 Mbps (theoretical speed) - within 30 m Range
* 4-6 Mbps (actual speed)
* 100 -150 feet range
* Most popular, Least Expensive
* Interference from mobile phones and Bluetooth devices which can reduce the transmission speed.

**IEEE 802.11a:**

* Introduced in 2001
* Operates at 5 GHz (less popular)
* 54 Mbps (theoretical speed)
* 15-20 Mbps (Actual speed)
* 50-75 feet range
* More expensive
* Not compatible with 802.11b

**IEEE 802.11g:**

* Introduced in 2003
* Combine the feature of both standards (a, b)
* 100-150 feet range
* 54 Mbps Speed
* 2.4 GHz radio frequencies
* Compatible with ‘b’

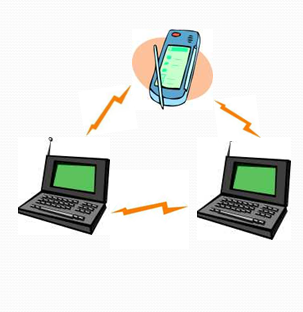
**Elements of a WI-FI Network:**

* Access Point (AP) - The AP is a wireless LAN transceiver or “base station” that can connect one or many wireless devices simultaneously to the Internet.
* Wi-Fi cards - They accept the wireless signal and relay information. They can be internal and external.
* Safeguards - Firewalls and anti-virus software protect networks from uninvited users and keep information secure.

**Wi-Fi Network Topologies:**

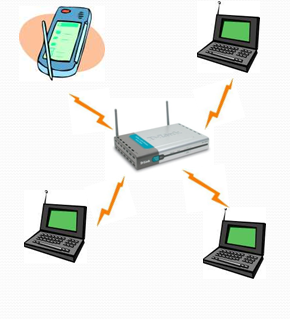
* Peer-to-peer topology (Ad-hoc Mode)
* AP-based topology (Infrastructure Mode)

**Peer-to-peer topology:**

****

* AP is not required.
* Client devices within a cell can communicate with each other directly.
* It is useful for setting up a wireless network quickly and easily.

**Infrastructure network:**

****

* The client communicate through Access Point.
* Any communication has to go through AP.
* If a Mobile Station (MS), like a computer, a PDA, or a phone, wants to communicate with another MS, it needs to send the information to AP first, then AP sends it to the destination MS.

**Hotspots:**

* A Hotspot is a geographical area that has a readily accessible wireless network
* Hotspots are equipped with Broad band Internet connection and one or more Access points that allow users to access the internet wirelessly
* Hotspots can be setup in any public location that can support an Internet connection.
* Ex : Airport, Railway station, Bus station etc.

**How a Wi-Fi Network Works?**

* A Wi-Fi hotspot is created by installing an access point to an internet connection.
* An access point acts as a base station.
* When Wi-Fi enabled device encounters a hotspot the device can then connect to that network wirelessly.
* A single access point can support up to 30 users and can function within a range of 100 – 150 feet indoors and up to 300 feet outdoors.
* Many access points can be connected to each other via Ethernet cables to create a single large network.

**Advantages:**

* Mobility
* Ease of Installation
* Flexibility
* Cost
* Reliability
* Security
* Use unlicensed part of the radio spectrum
* Roaming
* Speed

**SOFTWARE DEVELOPMENT**

**Programming**

The Arduino/Genuino Uno can be programmed with the ([Arduino Software](https://www.arduino.cc/en/Main/Software) (IDE)). Select "Arduino/Genuino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the [reference](https://www.arduino.cc/en/Reference/HomePage) and [tutorials](https://www.arduino.cc/en/Tutorial/HomePage).

The ATmega328 on the Arduino/Genuino Uno comes preprogrammed with a [bootloader](https://www.arduino.cc/en/Hacking/Bootloader?from=Tutorial.Bootloader) that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol ([reference](http://www.atmel.com/Images/doc2525.pdf), [C header files](http://www.atmel.com/dyn/resources/prod_documents/avr061.zip)).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using [Arduino ISP](https://www.arduino.cc/en/Main/ArduinoISP) or similar; see [these instructions](https://www.arduino.cc/en/Hacking/Programmer) for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

* On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then rese ing the 8U2.
* On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use [Atmel's FLIP software](http://www.atmel.com/products/microcontrollers/default.aspx) (Windows) or the [DFU programmer](http://dfu-programmer.github.io/) (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See [this user-contributed tutorial](http://forum.arduino.cc/index.php/topic,111.0.html) for more information.

**Warnings**

The Arduino/Genuino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. 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.

**Differences with other boards**

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

**Power**

The Arduino/Genuino Uno board 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 (wall-wart) 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 from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

* Vin. The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
* 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
* 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
* GND. Ground pins.
* IOREF. This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

**Memory**

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the [EEPROM library](https://www.arduino.cc/en/Reference/EEPROM)).

**Input and Output**

Each of the 14 digital pins on the Uno can be used as an input or output, using [pinMode()](https://www.arduino.cc/en/Reference/PinMode), [digitalWrite()](https://www.arduino.cc/en/Reference/DigitalWrite), and [digitalRead()](https://www.arduino.cc/en/Reference/DigitalRead) functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

* Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
* External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
* PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
* SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
* LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
* TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference()function.  
There are a couple of other pins on the board:

* AREF. Reference voltage for the analog inputs. Used with analogReference().
* Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

**Communication**

**-**

Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](https://www.arduino.cc/en/Guide/Windows#toc4). The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [Software Serial library](https://www.arduino.cc/en/Reference/SoftwareSerial) allows serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the I2C bus; see the [documentation](https://www.arduino.cc/en/Reference/Wire) for details. For SPI communication, use the [SPI library](https://www.arduino.cc/en/Reference/SPI).

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino/Genuino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino Software (IDE) uses this capability to allow you to upload code by simply pressing the upload button in the interface toolbar. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno board contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see [this forum thread](http://forum.arduino.cc/index.php/topic,22974.0.html) for details.

Arduino programs may be written in any [programming language](https://en.wikipedia.org/wiki/Programming_language) with a compiler that produces binary machine code. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio, which can be used for programming Arduino.

The Arduino project provides the Arduino [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE), which is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) application written in the programming language [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It originated from the IDE for the languages [*Processing*](https://en.wikipedia.org/wiki/Processing_(programming_language)) and [*Wiring*](https://en.wikipedia.org/wiki/Wiring_(development_platform)). It was created for people with no profound knowledge of electronics. It includes a code editor with features such as [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting), [brace matching](https://en.wikipedia.org/wiki/Brace_matching), cutting-pasting and searching-replacing text, and automatic indenting, and provides simple one-click mechanism to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a series of menus.

A program written with the IDE for Arduino is called a "sketch".[[40]](https://en.wikipedia.org/wiki/Arduino#cite_note-43) Sketches are saved on the development computer as files with the file extension .ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension .pde.

The Arduino IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B) using special rules to organize code. The Arduino IDE supplies a [software library](https://en.wikipedia.org/wiki/Software_library) from the [Wiring](https://en.wikipedia.org/wiki/Wiring_(development_platform)) project, which provides many common input and output procedures. User-written code only requires two functions, for starting the sketch and the main programs loop, that are compiled and linked with a program stub *main()* into an executable [cyclic executive](https://en.wikipedia.org/wiki/Cyclic_executive) program with the [GNU toolchain](https://en.wikipedia.org/wiki/GNU_toolchain), also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal coding that is loaded into the Arduino board by a loader program in the board's firmware

A minimal Arduino C/C++ sketch, as seen by the Arduino IDE programmer, consist of only two functions:[[41]](https://en.wikipedia.org/wiki/Arduino" \l "cite_note-44)[[42]](https://en.wikipedia.org/wiki/Arduino#cite_note-45)[[43]](https://en.wikipedia.org/wiki/Arduino#cite_note-46)

* *setup()*: This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.
* *loop()*: After setup() is called, this function is called repeatedly by a program loop in the main program. It controls the board until it is powered off or is reset.

Most Arduino boards contain a [light-emitting diode](https://en.wikipedia.org/wiki/Light-emitting_diode) (LED) and a load resistor connected between pin 13 and ground, which is a convenient feature for many tests and program functions.[[44]](https://en.wikipedia.org/wiki/Arduino#cite_note-Blink_Tutorial-47) A typical program for a beginning Arduino programmer blinks an LED repeatedly. This program is usually loaded in the Arduino by the manufacturer. In the Arduino environment, a user might write such a program as shown:

#define LED\_PIN 13 // Pin number attached to LED.

void setup() {

pinMode(LED\_PIN, OUTPUT); // Configure pin 13 to be a digital output.

}

void loop() {

digitalWrite(LED\_PIN, HIGH); // Turn on the LED.

delay(1000); // Wait 1 second (1000 milliseconds).

digitalWrite(LED\_PIN, LOW); // Turn off the LED.

delay(1000); // Wait 1 second.

}

This program uses the functions *pinMode()*, *digitalWrite()*, and *delay()*, which are provided by the internal libraries included in the IDE environment.

**IOT**

IoT (Internet of Things) is an advanced automation and analytics system which exploits networking, sensing, big data, and artificial intelligence technology to deliver complete systems for a product or service. These systems allow greater transparency, control, and performance when applied to any industry or system.

IoT systems have applications across industries through their unique flexibility and ability to be suitable in any environment. They enhance data collection, automation, operations, and much more through smart devices and powerful enabling technology.

IoT systems allow users to achieve deeper automation, analysis, and integration within a system. They improve the reach of these areas and their accuracy. IoT utilizes existing and emerging technology for sensing, networking, and robotics.

IoT exploits recent advances in software, falling hardware prices, and modern attitudes towards technology. Its new and advanced elements bring major changes in the delivery of products, goods, and services; and the social, economic, and political impact of those changes.

IOT FEATURES

The most important features of IoT include artificial intelligence, connectivity, sensors, active engagement, and small device use. A brief review of these features is given below −

* AI − IoT essentially makes virtually anything “smart”, meaning it enhances every aspect of life with the power of data collection, artificial intelligence algorithms, and networks. This can mean something as simple as enhancing your refrigerator and cabinets to detect when milk and your favorite cereal run low, and to then place an order with your preferred grocer.
* Connectivity − New enabling technologies for networking, and specifically IoT networking, mean networks are no longer exclusively tied to major providers. Networks can exist on a much smaller and cheaper scale while still being practical. IoT creates these small networks between its system devices.
* Sensors − IoT loses its distinction without sensors. They act as defining instruments which transform IoT from a standard passive network of devices into an active system capable of real-world integration.
* Active Engagement − Much of today's interaction with connected technology happens through passive engagement. IoT introduces a new paradigm for active content, product, or service engagement.
* Small Devices − Devices, as predicted, have become smaller, cheaper, and more powerful over time. IoT exploits purpose-built small devices to deliver its precision, scalability, and versatility.

IOT ADVANTAGES

The advantages of IoT span across every area of lifestyle and business. Here is a list of some of the advantages that IoT has to offer −

* Improved Customer Engagement − Current analytics suffer from blind-spots and significant flaws in accuracy; and as noted, engagement remains passive. IoT completely transforms this to achieve richer and more effective engagement with audiences.
* Technology Optimization − The same technologies and data which improve the customer experience also improve device use, and aid in more potent improvements to technology. IoT unlocks a world of critical functional and field data.
* Reduced Waste − IoT makes areas of improvement clear. Current analytics give us superficial insight, but IoT provides real-world information leading to more effective management of resources.
* Enhanced Data Collection − Modern data collection suffers from its limitations and its design for passive use. IoT breaks it out of those spaces, and places it exactly where humans really want to go to analyze our world. It allows an accurate picture of everything.

IOT SOFTWARE

IoT software addresses its key areas of networking and action through platforms, embedded systems, partner systems, and middleware. These individual and master applications are responsible for data collection, device integration, real-time analytics, and application and process extension within the IoT network. They exploit integration with critical business systems (e.g., ordering systems, robotics, scheduling, and more) in the execution of related tasks.

## Data Collection

This software manages sensing, measurements, light data filtering, light data security, and aggregation of data. It uses certain protocols to aid sensors in connecting with real-time, machine-to-machine networks. Then it collects data from multiple devices and distributes it in accordance with settings. It also works in reverse by distributing data over devices. The system eventually transmits all collected data to a central server.

## Device Integration

Software supporting integration binds (dependent relationships) all system devices to create the body of the IoT system. It ensures the necessary cooperation and stable networking between devices. These applications are the defining software technology of the IoT network because without them, it is not an IoT system. They manage the various applications, protocols, and limitations of each device to allow communication.

## Real-Time Analytics

These applications take data or input from various devices and convert it into viable actions or clear patterns for human analysis. They analyze information based on various settings and designs in order to perform automation-related tasks or provide the data required by industry.

## Application and Process Extension

These applications extend the reach of existing systems and software to allow a wider, more effective system. They integrate predefined devices for specific purposes such as allowing certain mobile devices or engineering instruments access. It supports improved productivity and more accurate data collection.

IOT TECHNOLOGY AND PROTOCOLS

IoT primarily exploits standard protocols and networking technologies. However, the major enabling technologies and protocols of IoT are RFID, NFC, low-energy Bluetooth, low-energy wireless, low-energy radio protocols, LTE-A, and WiFi-Direct. These technologies support the specific networking functionality needed in an IoT system in contrast to a standard uniform network of common systems.

## NFC and RFID

RFID (radio-frequency identification) and NFC (near-field communication) provide simple, lowenergy, and versatile options for identity and access tokens, connection bootstrapping, and payments.

* RFID technology employs 2-way radio transmitter-receivers to identify and track tags associated with objects.
* NFC consists of communication protocols for electronic devices, typically a mobile device and a standard device.

## Low-Energy Bluetooth

This technology supports the low-power, long-use need of IoT function while exploiting a standard technology with native support across systems.

## Low-Energy Wireless

This technology replaces the most power hungry aspect of an IoT system. Though sensors and other elements can power down over long periods, communication links (i.e., wireless) must remain in listening mode. Low-energy wireless not only reduces consumption, but also extends the life of the device through less use.

## Radio Protocols

ZigBee, Z-Wave, and Thread are radio protocols for creating low-rate private area networks. These technologies are low-power, but offer high throughput unlike many similar options. This increases the power of small local device networks without the typical costs.

## LTE-A

LTE-A, or LTE Advanced, delivers an important upgrade to LTE technology by increasing not only its coverage, but also reducing its latency and raising its throughput. It gives IoT a tremendous power through expanding its range, with its most significant applications being vehicle, UAV, and similar communication.

## WiFi-Direct

WiFi-Direct eliminates the need for an access point. It allows P2P (peer-to-peer) connections with the speed of WiFi, but with lower latency. WiFi-Direct eliminates an element of a network that often bogs it down, and it does not compromise on speed or throughput.

BLYNK APP

Blynk is a toolset for all makers, badass inventors, designers, teachers, nerds and geeks who would love to use their smartphones to control electronics like Arduino, RaspberryPi and similar ones. We’ve done all the hard work of establishing internet connection, building an app and writing hardware code.

With Blynk, you simply snap together an amazing interface from various widgets we provide, upload the example code to your hardware and enjoy seeing first results in under 5 minutes! It works perfectly for newbie makers and saves tons of time for evil geniuses.

Blynk will work with all popular boards and shields. We wanted to give you full freedom when deciding how to plug Blynk into your existing or new project. You will also enjoy the convenience of Blynk Cloud. Which is, by the way is free and open-source.

Imagine a prototyping board on your smartphone where you drag and drop buttons, sliders, displays, graphs and other functional widgets. And in a matter of minutes these widgets can control Arduino and get data from it.

Blynk is not an app that works only with a particular shield. Instead, it's been designed to support the boards and shields you are already using. And it works on iOs and Android.

UPD: Blynk also works over USB. This means you can tinker with the app by connecting it to your laptop or desktop while waiting for some internet shield to arrive.

Blynk works over the Internet. So the one and only requirement is that your hardware can talk to the Internet.

No matter what type of connection you choose - Ethernet, Wi-Fi or maybe this new ESP8266 everyone is talking about – Blynk libraries and example sketches will get you online, connect to Blynk Server and pair up with your smartphone.

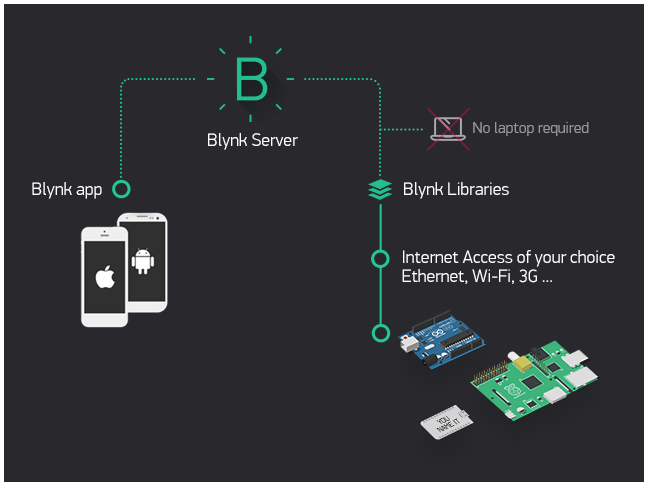


FIG: Blynk architecture

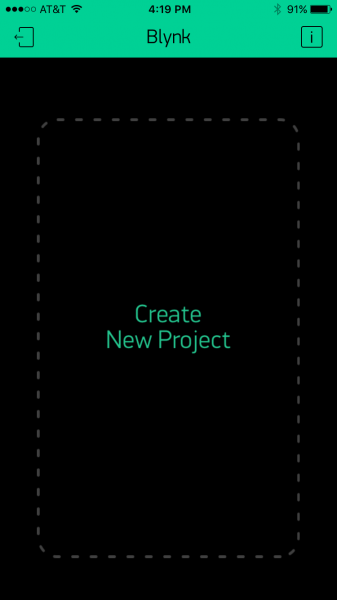
Currently, Blynk libraries work with:

* USB
* Ethernet shield
* WiFi shield
* Arduino with Ethernet
* Arduino YÚN (testing in progress)
* ESP8266
* Raspberry Pi (Blynk will communicate with Pi's GPIOs)
* more Arduino compatible shields and boards (this list will be updated as we test the compatibility)

It's not that easy to take Arduino out of your home network, so we've built a Blynk server. It handles all the authentication and communication, and also keeps an eye on your board while the smartphone is offline. Blynk server runs on Java and is open-source. You will be able to run it locally if you really need to. Messaging between mobile apps , Blynk Server and Arduino is based on a simple, lightweight and fast binary protocol over TCP/IP sockets.

CREATING A PROJECT IN BLYNK APP

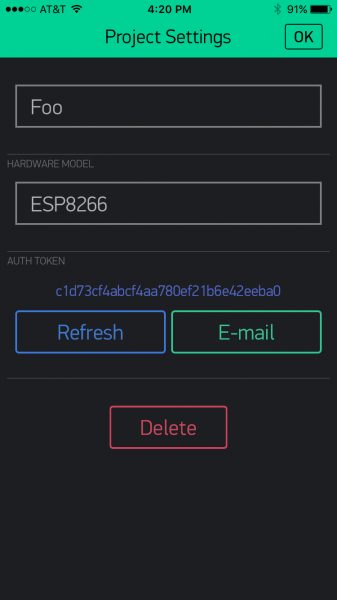
After downloading the app, create an account and log in. Welcome to Blynk!

[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/blynk-blank.PNG)

You’ll also need to install the **Blynk Arduino Library**, which helps generate the firmware running on your ESP8266. Download the latest release from Blynk’s GitHub repo, and follow along with the directions there to install the required libraries.

### Create a Blynk Project

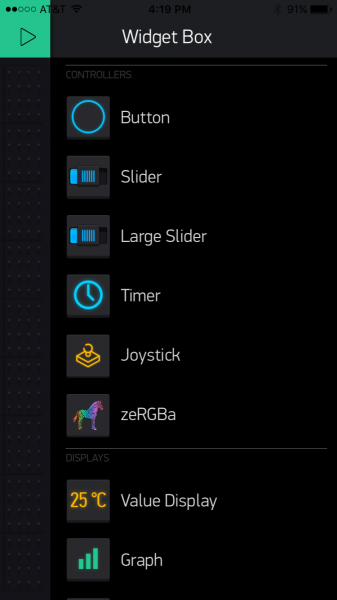
Next, click the “Create New Project” in the app to create a new Blynk app. Give it any name you please, just make sure the “Hardware Model” is set to **ESP8266**.

[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/Blynk-new.PNG)

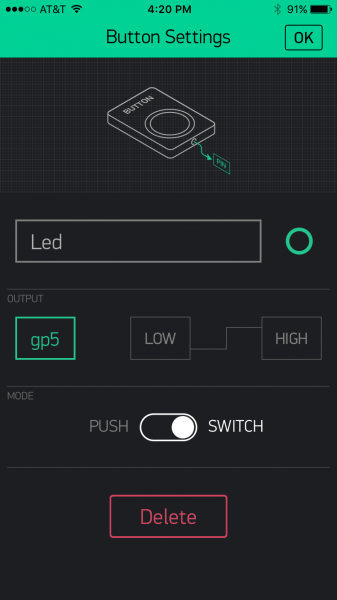
The **Auth Token** is very important – you’ll need to stick it into your ESP8266’s firmware. For now, copy it down or use the “E-mail” button to send it to yourself.

### Add Widgets to the Project

Then you’ll be presented with a blank new project. To open the widget box, click in the project window to open.

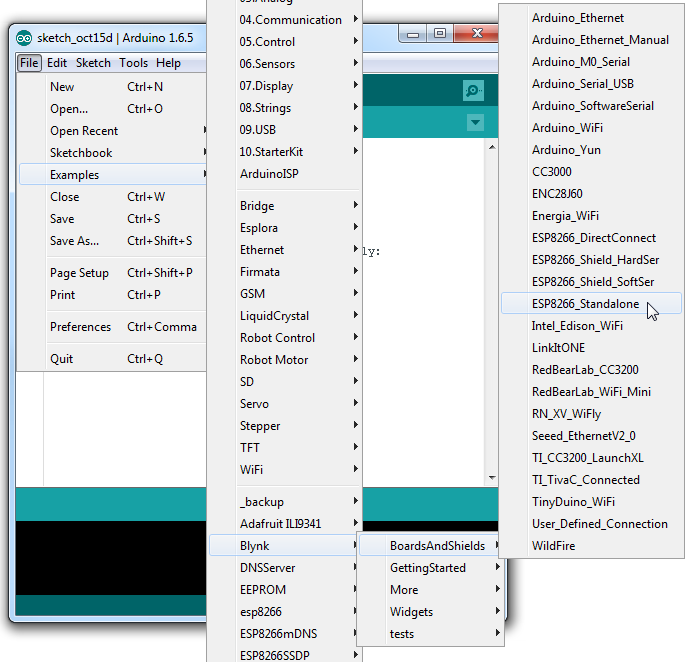
[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/Blynk-widgetBox.PNG)

Add a **Button**, then click on it to change its settings. Buttons can toggle outputs on the ESP8266. Set the button’s output to **gp5**, which is tied to an LED on the Thing Dev Board. You may also want to change the action to “Switch.”

[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/Blynk-Button.PNG)

### Upload the Blynk Firmware

Now that your Blynk project is set up, open Arduino and navigate to the **ESP8266\_Standalone** example in the **File**>**Examples**>**Blynk**>**BoardsAndShields** menu.

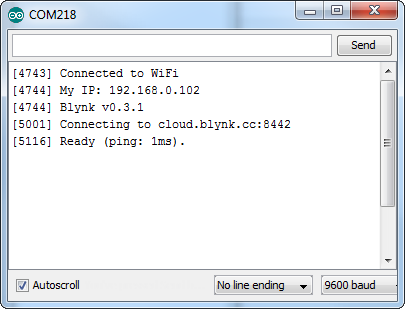
[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/Blynk-example.png)

Before uploading, make sure to paste your **authoriazation token** into the auth[] variable. Also make sure to **load your WiFi network settings into the** Blynk.begin(auth, "ssid", "pass") function.

Then upload!

### Run the Project

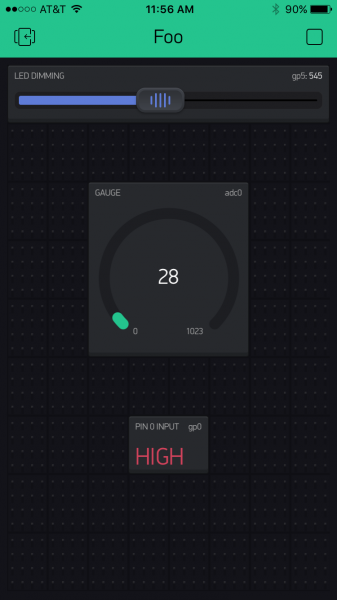
After the app has uploaded, open the serial monitor, setting the baud rate to 9600. Wait for the “Ready (ping: xms).” message.

[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/Blynk-Serial.png)

Then click the “Run” button in the top right corner of the Blynk app. Press the button and watch the LED!

[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/esp8266-blynk.jpg)

Then add more widgets to the project. They should immediately work on the ESP8266 without uploading any new firmware.

[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/Blynk-fullExample.PNG)

You can add analog output sliders, digital input monitors, and analog input gauges.

**Advantages:**

## Reliable Food Source

* can be combined with irrigation practices

## Availability OF FISH SOURCES

* Planning the reduction of external technical assistance, so that farmers take complete control of their production projects

**Applications:**

* energy reliability, management
* communication security
* data processing sensor status in aquaculture maintenance of when sensor data changes

**CONCLUSION**

* Developing the capacity of farmers so that, through their own observations and reasoning, they are able to solve limitations to the implementation of rural aquaculture projects, thereby improving the productivity of household farm units
* Protecting water sources.
* Generating a reliable data base on production and the costs of different production systems.
* Achieving the capacity of farmers to market their agricultural products, including those of aquaculture origin, which contribute to income generation.

**Reference:**

1. S.Kayalvizhi, Koushik Reddy G, Vivek Kumar P, VenkataPrasanth N “Cyber Aqua Culture Monitoring System Using ArdunioAnd Raspberry Pi,” International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 4, Issue 5,Pg:2320-3765; May 2015.
2. [2] Suresh Babu Chandanapalli,Sreenivasa Reddy E and Rajya Lakshmi D, “Design and Deployment of Aqua Monitoring System Using Wireless Sensor Networks and IAR-Kick,” Jouranl of Aquaculture Research&Development ,Volume 5 • Issue 7 • 1000283
3. Wen-TsaiSung,Jui-HoChen,Da-ChiunHuang,Yi-HaoJu,“MultisensorsRealtime Data Fusion Optimization for IOT Systems,” 2014 IEEE International Conference on Systems, Man, and Cybernetics October 5-8, 2015, San Diego,CA,USA.
4. WeerasakCheunta, NitthitaChirdchoo and KanitthaSaelim, “Efficiency Improvement of an Integrated Giant Freshwater-White Prawn Farming in Thailand Using a Wireless Sensor Network,” Research Gate,DECEMBER2014.