Arduino Development Environment

The Arduino development environment contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

Writing Sketches

Software written using Arduino are called sketches. These sketches are written in the text editor. Sketches are saved with the file extension .ino. It has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino environment including complete error messages and other information. The bottom righthand corner of the window displays the current board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the IDE prior to 1.0 saved sketches with the extension pde It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

The Arduino environment uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

'''Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

Tabs, Multiple Files, and Compilation

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no extension), C files (.c extension), C++ files (.cpp), or header files (.h).

Uploading

Before uploading your sketch, you need to select the correct items from the Tools Board and Tools Serial Portmenus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241(for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or/dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyUSB0,/dev/ttyUSB1 or similar.

Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the File menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino environment will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch Import Library menu. This will insert one or more  statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete it from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

PROGRAMMING

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the Tools Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a 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, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . 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 resetting 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 (Windows) or the DFU programmer (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 for more information.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of theATmega8U2/16U2 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 uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. 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 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 for details.

STEP 1: Building of sensor network

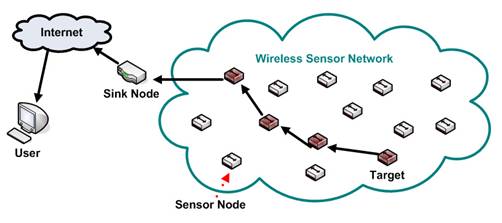


Figure 3.1: Conventional Sensor Network

A conventional sensor network is a radio network of sensor nodes with ability to sense physical parameters, store sensed data, carry out simple processing on data and forward the data through radio interface. The objective of such network is to push the data to a sink node which can then forward the data to server ( or cloud) is shown in the figure 3.1.

STEP 2: Connection of sensor network to the cloud.

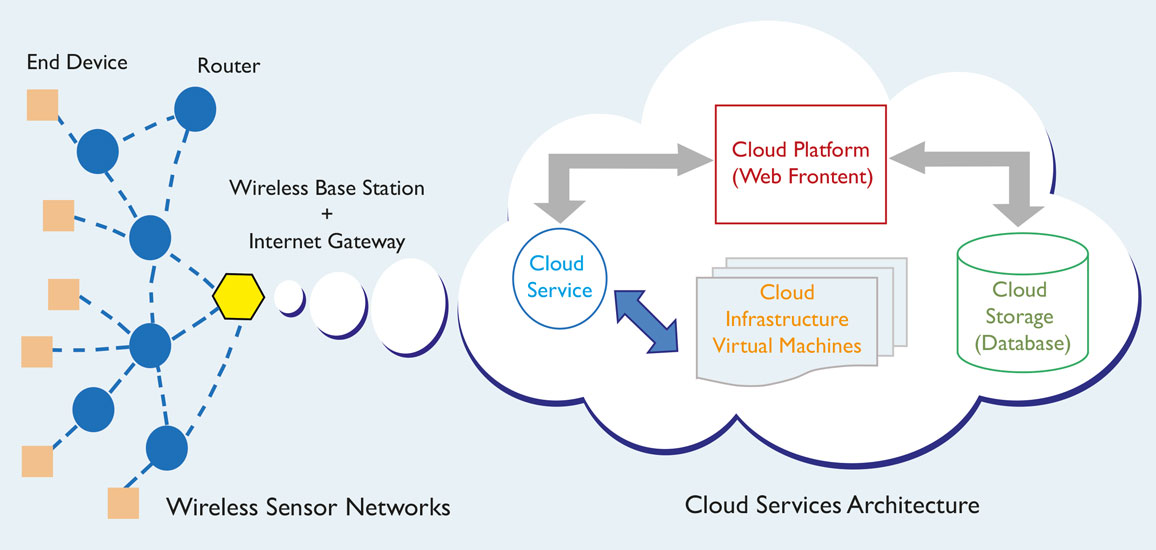


Figure 3.2: Sensor Network over cloud

Figure 3.2 shows the connection between sensor network and the cloud. However many real time applications includes sensors spread over long areas. As such they are treated as independent networks. Internet of Things is a new paradigm of connecting devices like micocontrollers and smart objects to cloud. Using IoT services, we can now connect sensors to internet directly. One of common design of sensor network includes cluster based methods where clusters are at formed by group of nodes. These are also called coordinator nodes. These nodes gather data from all neihboring nodes. If these nodes can be linked to internet with their unique Ip addresses, then the sensor network can be infinitely scaled( theoretically).

STEP 3: Interfacing with the embedded system.

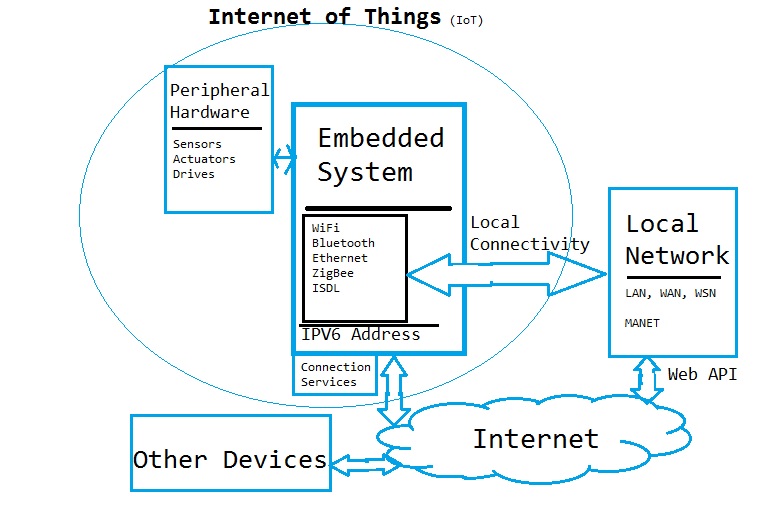


Figure 3.3: Structure Of IOT

The basic structure of IoT is presented above. Our work would include following innovative research extensions to the exising WSN and IoT framework.

Firstly we would focus purely on sensors and that too on coordinator nodes in Peripheral hardware. Rather than working on integrating individual hardware over cloud, our method would assume entire standalone sensor network as a single peripheral and would connect that to cloud though IoT. We would focus mainly on ZigBee as wireless technology as that is most accepted WSN standard.Our methods would provide not only communication services but also data gathering and analysis services. We would integrate both simulation as well as real time test beds to prove the designed concepts.

STEP 4: Possible outcome of the IoT:

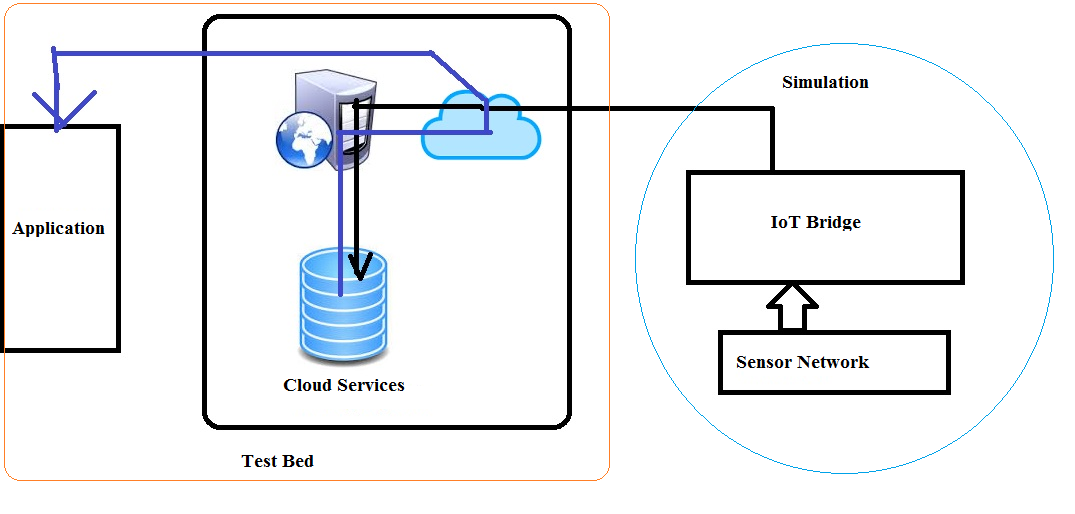


Figure 3.4: possible outcome of project

In the project work we mainly focus on building more comprehensive state of art cloud extension of WSN through IoT. Firstly the research would focus towards bettering each of the current state of art building blocks including but not limited to sensor network, coordinator protocol, data analysis in sensor network, cloud services, IoT protocols and so on. One of the first expected result would be a unique framework to connect existing WSN to cloud. Then system should prove the advantage of such extension by demonstrating the scale of improvement in data analysis services .Results should prove that IoT can be used to create mesh sensor networks and enhanced bandwidth can be used to connect sensor networks with other control system.

All the information given by the sensor network should be passed to the cloud through the javascriptnode.js . Here the cloud is a private application which can be accessed by the public and this application can be easily installed by any user which will helps the user to collect the data from the sensor network and also helps to analyze and visualize the data from the sensor network . This can be done by the matlab visualization which build in option provided in this apps. According to the data given by the apps the user can act based on the requirements ..

STEP 5: Sending data to THING SPEAK APPS:

The [Internet of Things](http://www.mathworks.com/solutions/internet-of-things/) provides access to a broad range of embedded devices and web services. [Thing Speak](http://www.thingspeak.com/) is an open data platform and API for the Internet of Things that enables us to collect, store, analyze, visualize, and act on data from sensors or actuators, such as Arduino®, and other hardware. For example, with Thing Speak one can create sensor-logging applications, location-tracking applications, and a social network of things with status updates, so that you could have your home thermostat control itself based on our current location.

The primary element of Thing Speak activity is the channel, which contains data fields, location fields, and a status field. After you create a Thing Speak channel, you can write data to the channel, process and view the data with MATLAB® code, and react to the data with tweets and other alerts.

The typical Thing Speak workflow lets you:

Create a [Channel](https://thingspeak.com/channels) and collect data

[Analyze](https://thingspeak.com/apps/matlab_analyses) and [Visualize](https://thingspeak.com/apps/matlab_visualizations) the data

Act on the data using any of several [Apps](https://thingspeak.com/apps)