Unit 5: Prototyping Connected (Embedded) Devices for the Internet of Things(IoT) / Machine to Machine(M2M)

By-Er. Ishwar Rathod
SCSIT, SUAS, Indore.

Prototyping Embedded Software on Arduino /on Arduino boards

Prototyping Embedded Software

- Develop the codes, design and test the embedded devices for IoT and M2M using the IDEs and development platforms

First and Second levels in IoT architectural concept

- Gathering (data from devices/sensors) + consolidating (enriching)
- Connection to Internet

Use of IDE

- An IDE enables development of software for functions at first and second levels
- IDE may also enable usages of the OS or RTOS functions at an embedded device

Bootloader firmware and IDE

- Stores at flash/ROM of microcontroller in a device and enables communication with computer having an IDE
- The IDE, in general, consists of the APIs, libraries, compilers, RTOS, simulator, editor, assembler, debugger, emulators, logic analyser, code burner, and other software for integrated development of the system

IDE

- Enables the development of codes on a computer, and later on downloading
- (pushing) of codes on to embedded device, such as Arduino or microcontroller board
- A code-burner places codes into flash memory or EEROM or EPROM
- The specific application codes thus embed into the device

Arduino board programming

- The Arduino board has a pre-installed bootloader embedded into firmware
- Program development in C++ using avr-gcc tools
- Arduino programmer develops the codes using a graphical cross-platform IDE.

Arduino board programming

- Arduino provides simplicity
- IDE of Arduino board simplicity of beed of two functions
- Based on processing language

Arduino board programming

- The board connects to a computer which runs the IDE
- Bootloader program handovers the control and enables running of loader, which loads the required OS functions and software into the system hardware and networking capabilities into the board

The Arduino Bootloader Provisions For Multitasking

- Usage of interrupt (analogous to eventing) handing functions for each task
- Multitasking done by assigning multiple values of a number n for the tasks (n > 0)
- When an instruction for interrupt, INT n executes, then interrupt-handing function *n* is called for execution
- Each task or thread can have the number n associated with it

The Arduino IDE Provisions For Multitasking

- First, a computer downloads an appropriate IDE version, as per the computer OS.
- A computer usually runs Windows or Mac OS X or Linux.
- The IDE consists of a set of software modules, which provide the software and hardware environment for developing and prototyping the software for the specific device platform.

The Arduino IDE

- Available from website of Arduino
- The Arduino IDE includes a C/C++ library
- The library is called Wiring for a project of the same name with open source module at a website.
- The Wiring library functions make coding easy for the Arduino IO operations

Simplicity of Arduino IDE

- Only two functions are necessary to define executable program functions for the board
- setup() and loop()
- The function setup() runs at the start and is used for initialising settings,
- function loop() has a program in endless loop using statement 'while (true) *{statements ;}' which runs till* power off.

Serial Monitor At The IDE

- Enables messages from the embedded software for microcontroller into the computer screen where IDE is setup
- The messages required during testing and debugging the downloaded software during test stages

Summary

We learnt

- Prototyping Embedded Software
- Bootloader facilitates handover the control and enables running of loader, which loads the required OS
- IDE enables development of software and use of the OS
- IDE available from website of Arduino

Summary

We learnt

- Two functions simplicity; Setup ( ) and loop ( )
- Serial Monitor provides the messages required during testing and debugging the downloaded software during test stages

Contents

- 1) Trends of implementation of IoT applications
  - REST
  - Cloud
- 2) Connected-device Prototyping Tools
  - Arduino
  - Raspberry Pi
  - Intel Edison
  - Gadgeteer
- 3) Building Web-Connected Devices With Gadgeteer
  - #1. A simple camera
  - #2. A simple Internet webcam
  - #3. A sophisticated Web-controlled camera
  - #4. Logging sensor data using Cloud-based storage
  - #5. OCR using cloud-based processing
- 4) Comparison and Recommendation

Introduction

- IoT vision
  - Internet connectivity extends to the very simplest electronic devices (things)(with a IPv6 address?)
- Advantages with IoT
  - Our productivity can be improved when the things involved in our daily activities become “alive” or manageable.
- Types of IoT
  - Networked versions of commonplace devices
    - Refrigerators, TV, toaster, alarm clocks, doorbells, and so on
  - Embedded devices
    - Allow applications with simple electronic devices.
    - Numerous kinds of product exist!

1) Trends of implementation of IoT applications

- Using REST as a command protocol for web-to-serial applications
  - Cloud-based service (REST-ful)

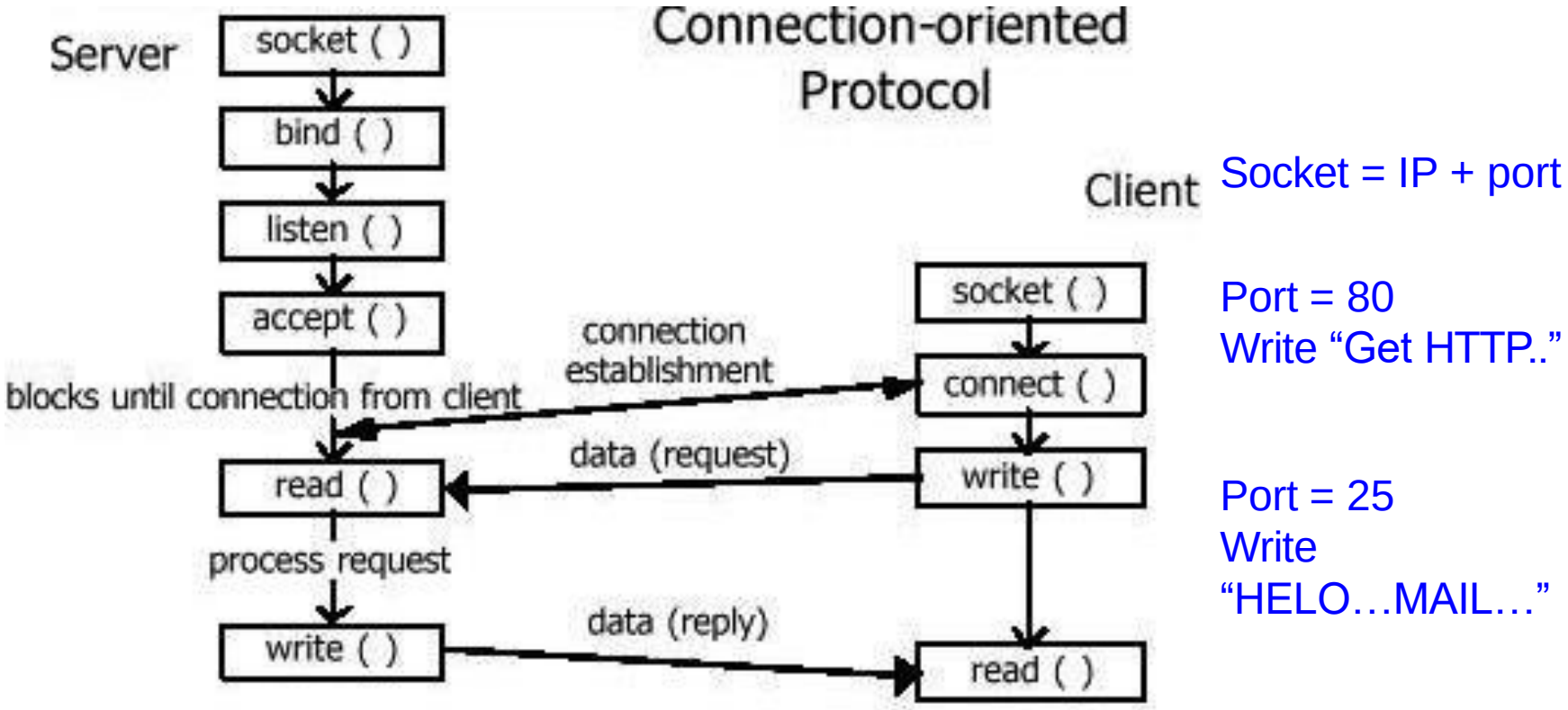
Cloud-based Web services

- The software running within the embedded device will increasingly be complemented by cloud-based Web services.
- Dramatically extend the effective processing and storage capabilities of these connected devices
- New class of applications might emerge
  - Groups of device act as the I/O elements of potentially global-scale distributed services and applications.

Cloud-based Web services

Using REST as a command protocol for web-to-serial applications

- Socket programming



Socket = IP + port

Port = 80
Write “Get HTTP..”

Port = 25
Write
“HELO…MAIL…”

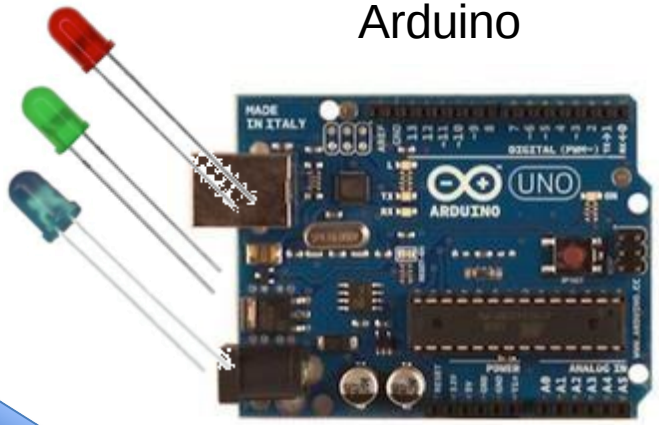
Using REST as a command protocol for web-to-serial applications

- Representational State Transfer (REST)
- Conventional use URL
  - A URL refers to a particular document on a server.
    - http://www.ouhk.edu.hk/prospectus.html
- REST (Representational State Transfer)
  - A URL is used to **set or get the state** of web-based application.
  - To get the price of item 3045:
    - http://www.mystore.com/item/3045/price
  - To set the price of item 3045:
    - http://www.mystore.com/item/3045/price/4.99

Using REST as a command protocol for web-to-serial applications

- Existing Web frameworks make it possible for you to build a web server (in your embedded devices) that uses REST as the control protocol for your application.
  - Sinatra (for Ruby),
  - Flask (for Python) and
  - Express (for JavaScript through node.js)

Arduino



Web server and controller

sets the level of the LEDs (range from 0 to 100)

http://200.200.200.1/LED/r/18
http://200.200.200.1/LED/g/28
http://200.200.200.1/LED/b/8

node.js platform with express.js library

2) Introduction to Connected-device Prototyping Tools

Arduino, Raspberry Pi ,Intel Edison, Gadgeteer

Embedded platform for prototyping

Introduction to Embedded Electronics

- An **embedded** system is one kind of a computer system mainly designed to perform several tasks like to access, process, and store and also control the data in various **electronics**-based systems.

- **Embedded electronics** are specialized circuits that do something that wasn't possible with previous hardware. More often than not **embedded electronics** are stuffed into something that didn't have a computer in it before to make it more intelligent. Cars, refrigerators, planes, etc.

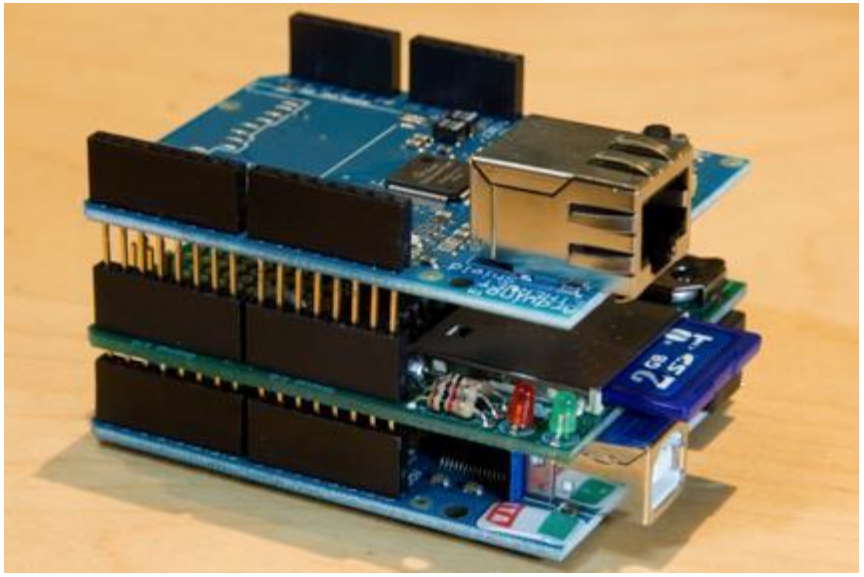
Arduino platform

- **What**
  - Microcontroller-based platforms
  - Started in 2005 in Italy, nowadays rather presenting a family of microcontroller boards rather than a specific one.
  - An internet magazine article from May 2011, stating that there were “about 300,000+ Arduino ‘in the wild’ at that time.

Arduino platform

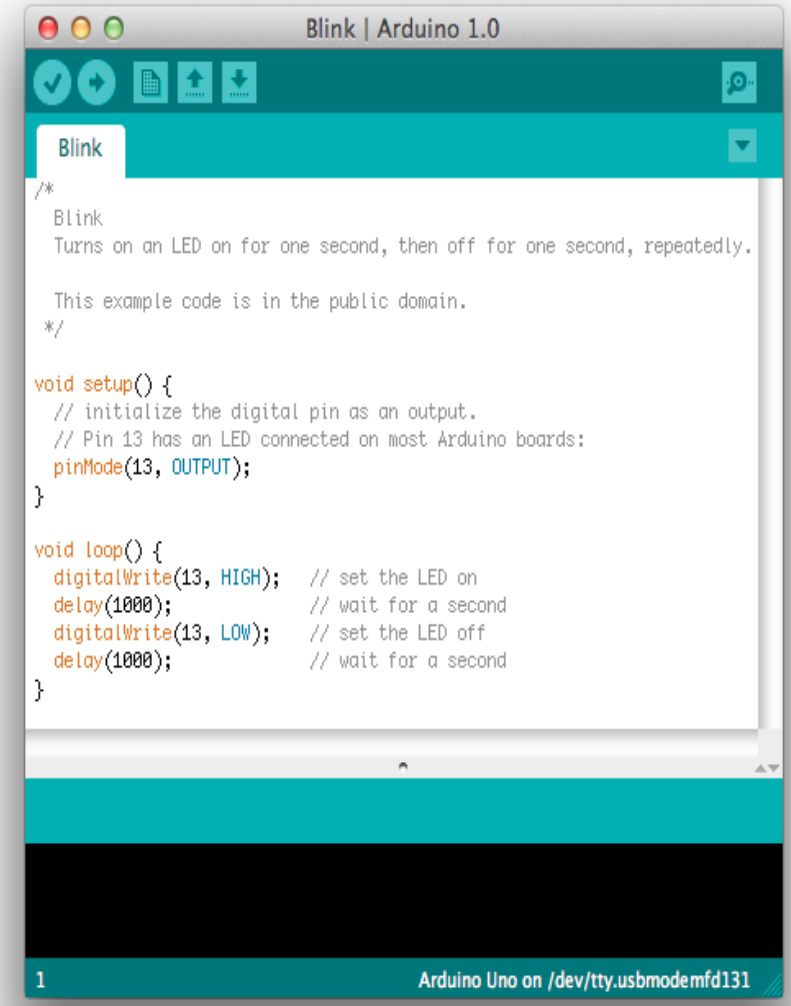
- **Hardware**
  - Shields -- add-on circuit boards that extend the platform's basic capabilities
  - Shields that provide Ethernet, Wi-Fi, and GPRS connectivity enable Arduino's use for connected-device development.

Arduino Uno



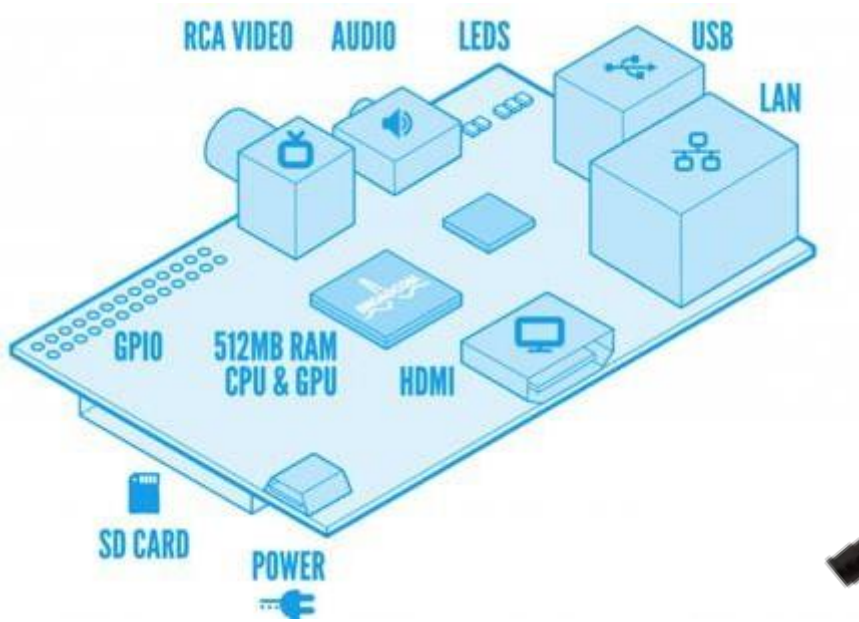
Arduino platform

- **IDE**
  - Minimalist integrated development environment (IDE)
  - Typically programmed with C
- **Debugging**
  - Debugging is typically supported via simple communications over a serial line interface
- **Software**
  - Developers commonly use the REST technique
    - because it is a lightweight, easy-to-debug way to communicate between connected devices
  - With REST, services are exposed and accessed using HTTP, which is readily supported by Arduino libraries that implement the relevant networking protocols and enable simple webserver operation.



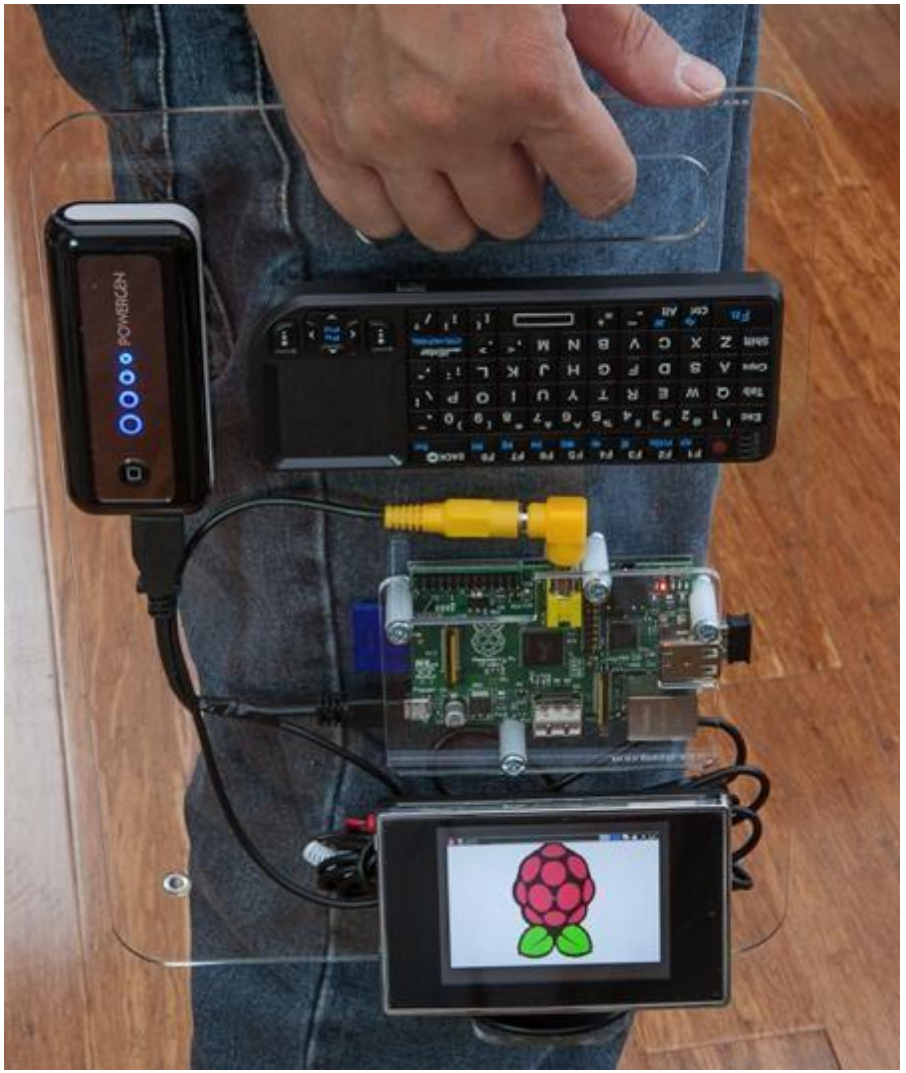
Raspberry Pi

- What
  - Small-form-factor Linux devices
  - Can be used for many of the things that your desktop PC does, like spreadsheets, word-processing, games, and playing high-definition video.



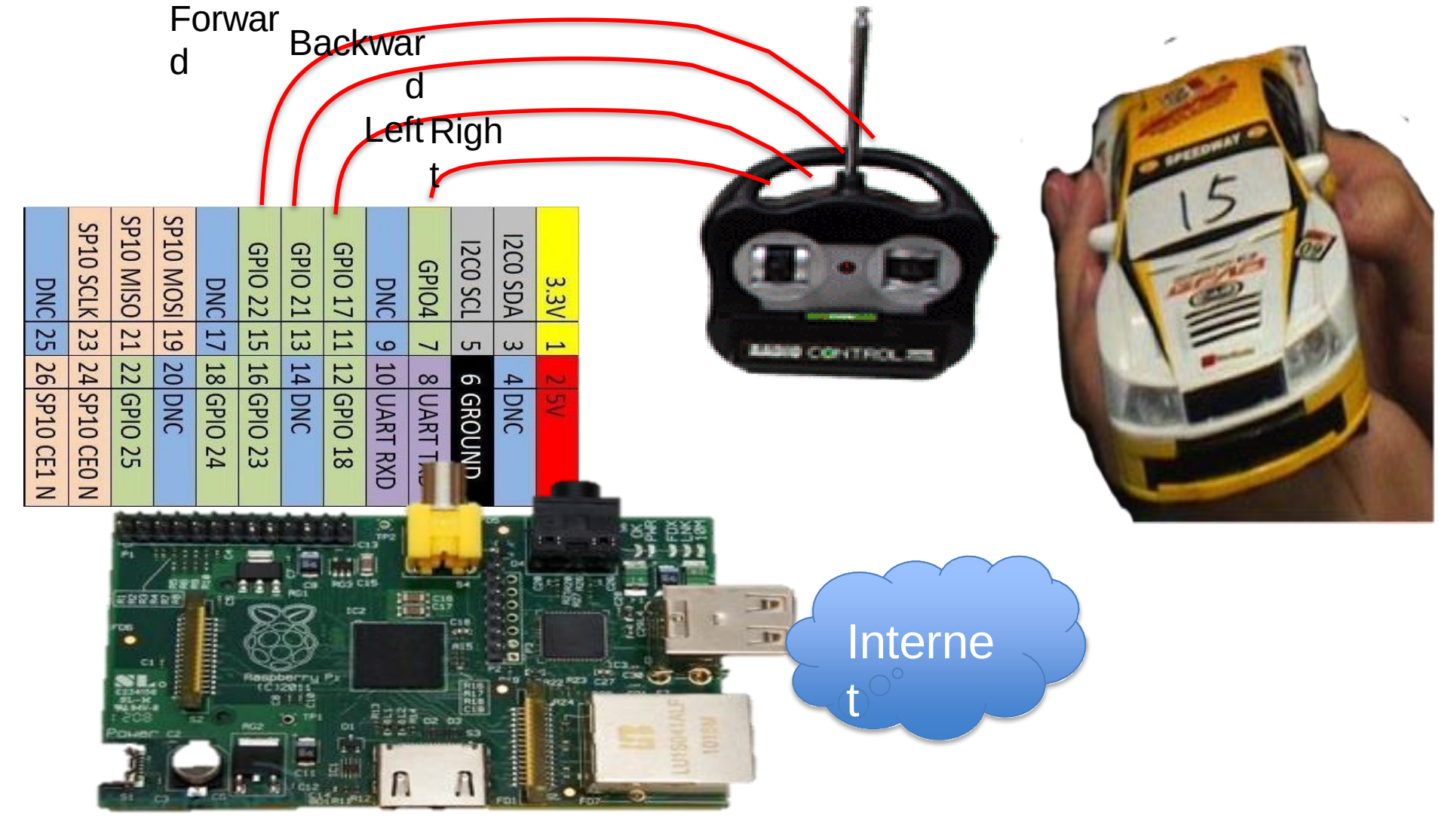
Raspberry Pi

- It uses standard off the shelf hardware.
  - The LCD is a low cost TFT monitor used in car reverse camera.
  - The battery pack is a standard portable USB charger.
  - Common wireless keyboard



Raspberry Pi

Raspberry Pi



Raspberry Pi

- Good
  - Offer the opportunity to leverage an extensive set of pre- existing tools and software components such as Node.js (http://nodejs. org),
    - which simplifies the implementation of REST-like asynchronous Web-based application programming interfaces (APIs).
  - Powerful and flexible
  - Fairly inexpensive: an ARM Linux box for US$25!
- Bad
  - Expose more complexity to the user
  - Typically less cost-effective than Arduino for lightweight device development.

IoT: Hardware, Middleware and Software

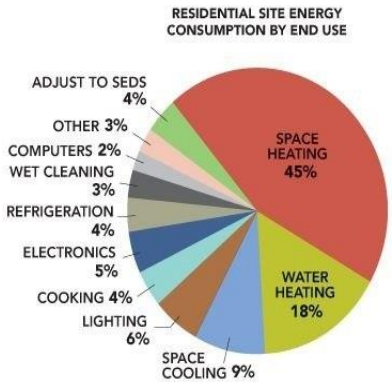
Hardware	Middleware	Software
Arduino Nano Pro Mini	OpenIoT	Eclipse IoT
Raspberry-Pi	OpenRemote	Google Brillo
IBM Watson	OpenHUB	IBM IoT Foundation
Azure	Kaa	Azure IoT Suit
AWS	Oracle-Fusion	Cloud Sensor
INTEL JOULE		Ninja Sphere
Netduino		Control Any
Flutter		Arduino

http://www.gadgetronicx.com/10-best-iot-hardware-platforms/

https://www.postscapes.com/internet-of-things-hardware/

http://www.gadgetronicx.com/10-best-iot-software-platforms/

Energy Saving, Monitoring and Security System

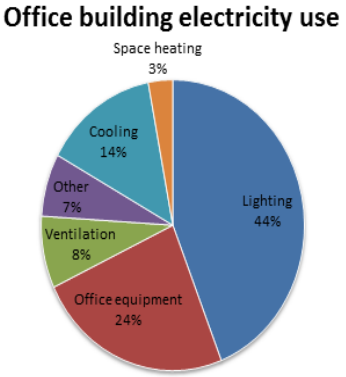


Incentive:

- Both residential and Office buildings waste a huge amount of electricity by virtue of overusing appliances due to different reasons like Ignorance, low maintenance or lack of awareness.
- We waste about 20-40% of residential and office building electricity due to this which makes up to 20% of the total energy produced.
- Most of this wastage can be prevented by using home and office appliances smartly. Since, time is of the essence for anyone such mistakes and inefficiency is only natural. Thus, the best solution is to automate their use.

Objectives:

- To engineer most energy efficient technique for automating the use of appliances in homes and offices that doesn't invade privacy, keeping security a paramount concern. We accomplished this by logically planning use of appliances on the basis of people present in the premises.
- We used infrared sensors for keeping track of people in the building instead of cameras which is both energy efficient and does not create a sense of being monitored all the time. Though we still have the capability of integrating a camera into the system if required.
- Some important features of the system are user defined profiles, authorized entry, security alarm and customizability and expandability.
- Future additions have also been left room for, from android integration, weather assessment and suggestion to remote locking and monitoring as it is designed using open source and



Result:

The System can detect entry and exit as well as the number of people with high accuracy in the premises and take actions like switching appliances on, off or set them at a given rate. Profiles according to residents can be created. It is also capable of detecting intrusion and alarming the correct authorities. We were able to save up to 70% of the wasted electricity in a medium sized setup.

Materials Required

For a prototype with floor plan including Two rooms, Four doors and Eight appliances

Raspberry Pi 3 model B & Micro SD cards with OS x1 **Rs. 2999/- & Rs. 500**

The heart of all the computation and interface for controlling the whole system, this is where logical decisions will be made based on gate sensor modules input, policies predetermined by the developer as well as a few policies built on user input and configuration.

Waveshare Raspberry Pi 3.5" Display Module x1 **Rs. 1900/-**

A screen for accessing the Pi directly without using any other system and while offline. Though it is optional but makes the system more portable and field programmable without connecting to insecure connections over LAN or Wi-Fi

Arduino UNO R3 x1 **Rs. 550/-**

For taking data from each gate module (NANO) and forwarding it to RPi 3

Arduino NANO V3.0 x4 **Rs. 300/-**

For taking analog input from combination of sensor modules of each gate and converting computing it to represent state of a gate and forwarding the data to central Arduino UNO which provides interface between analog input and Raspberry Pi.

Directional and Obstacle Sensor modules x20 **Rs. 35-55/-**

Directional: To find the direction in which any passing object moved. Based on principle of IR beam break detection.

Obstacle: To find which part of the gate was occupied at any instance of time. Also helps in recognizing whether the object is moving on legs or not.

Switching Module x2 **Rs. 200/-**

A decoder to convert incoming binary input into the index of relay which is to be switched on or off.

Miscellaneous: Sensors to facilitate decision making, elements to create circuits and basic electronic tools.

Heart of the Problem and Working Principal

Working Principal:

Any door can be treated as a half duplex mode of transaction of objects between two spaces.

Given the floor plan and ability to know the direction and number of such transactions we can compute which space is occupied by how many objects.

Solution and Application:

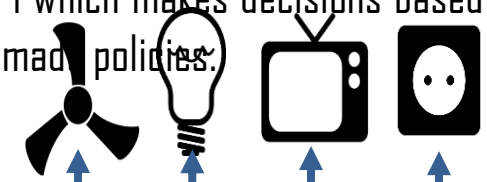
To build sensor modules which can identify in which direction an object passed and whether it had legs or not.

A pair of infrared beam breakers can be used to identify the direction of motion while another set of beam in coordination with the above mentioned sensor can identify if the body moved on legs by gap due to count of two legs per object.

Directional Sensors & Beam Break Sensors

Arduino Nano

- These Gate modules will be working in collaboration with each other to compute a desired output for this they will communicate to a Central Unit which will in turn take required actions.
- This Central Unit is the Raspberry Pi which makes decisions based on predetermined factors and user made policies.



Gate Module

DHT Module

UNO interface to RPi

Rpi 3/ Central Unit

Decoder and Relays

Code for Direction Sensor Module

Direction_update §

```
void setup() {// using A1 and A2 as input pins
  Serial.begin(9600);//initializing Serial port with baud rate=9600
}
//global variables to save the previous state of module
int prv=0,pa=0,pb=0,dir=0;
void loop() { //global variable to save the previous state of modul
  // a and b represent the two infrared pairs in the module
  int a=0,b=0;
  //reading the state of sensor a with threshold value as 920
  if(analogRead(A1)>920)
  a=1;
  else
  a=0;
  //reading the state of sensor b
  if(analogRead(A2)>920)
  b=1;
  else
  b=0;
  if(pa!=a)// checking for a change in state of sensor A
  {
    if(prv==2)
    dir--;// direction from B to A
    prv=1;
  }
  if(pb!=b)// checking for a change in state of sensor B
  {
    if(prv==1)
    dir++;// direction from A to B
    prv=2;
```

Direction_update §

```
  if(pa!=a)// checking for a change in state of sensor A
  {
    if(prv==2)
    dir--;// direction from B to A
    prv=1;
  }
  if(pb!=b)// checking for a change in state of sensor B
  {
    if(prv==1)
    dir++;// direction from A to B
    prv=2;
  }
  if(a==0)
  {
    if(b==0)
    {
      // when the sensor is free
      prv=0;
      // in case it is free just after object passed
      if(dir!=0)
      {
      Serial.println(dir);
      dir=0;
      }
    }
  }
  pa=a;// setting up the previous state for next iteration
  pb=b;
}
```

Code for DHT Sensor Module

DHT11

```
#include "dht.h"
#define dht_apin A0 // Analog Pin sensor is connected to

dht DHT;

void setup(){

  Serial.begin(9600);
  delay(500);//Delay to let system boot
  Serial.println("DHT11 Humidity & temperature Sensor\n\n");
  delay(1000);//Wait before accessing Sensor

}//end "setup()"

void loop(){
  //Start of Program

    DHT.read11(dht_apin);

    Serial.print("Current humidity = ");
    Serial.print(DHT.humidity);
    Serial.print("%  ");
    Serial.print("temperature = ");
    Serial.print(DHT.temperature);
    Serial.println("C  ");
    delay(5000);//Wait 5 seconds before accessing sensor again.
  //Fastest should be once every two seconds.
}// end loop()
```

Code for Interfacing gates to RPi

multiplexing_all_sensors_through_gates §

```
/*reading values from 4 gates having 3 sensor
 * modules each through a single arduino board*/
int gate1=2,gate2=5,gate3=8,gate4=11;
/*each value of gate represents the pin where
 * its first sensor module has been pinned*/
int sensors=3;
void setup() {
  /*using all 12 pins for representing sensor of gates*/
  for(int i=2;i<=13;i++)
  {
    pinMode(i,OUTPUT);
  }
  Serial.begin(9600);
  }
void loop() {
  readgate(gate1);
  readgate(gate2);
  readgate(gate3);
  readgate(gate4);//processing of the read data
  //setting outputs according to the set data
}
void readgate(int address)
{
  for(int i=0;i<sensors;i++)
  {
    digitalWrite(address+i,HIGH);
    digitalWrite(address+i,LOW);
  }
}
```

Embedded Computing

- Embedded computing systems combine hardware and software components that must work closely together.

- Embedded system designers have evolved design methodologies that play into our ability to embody part of the functionality of the system in software.

Characteristics of embedded computing applications

- Embedded computing is in many ways much more demanding than the sort of programs that you may have written for PCs or workstations.
- Functionality is important in both general-purpose computing and embedded computing, but embedded applications must meet many other constraints as well.
- On the one hand, embedded computing systems have to provide sophisticated functionality

sophisticated functionality

- Complex algorithms: The operations performed by the microprocessor/microcontroller may be very sophisticated. For example, the microprocessor/microcontroller that controls an automobile engine must perform complicated filtering functions to optimize the performance of the car while minimizing pollution and fuel utilization.

- User interface*:* Microprocessors/Microcontroller are frequently used to control complex user interfaces that may include multiple menus and many options. The moving maps in Global Positioning System (GPS) navigation are good examples of sophisticated user interfaces.

- To make things more difficult, embedded computing operations must often be performed to meet deadlines:

sophisticated functionality

- *Real time:* Many embedded computing systems have to perform in real time—if the data are not ready by a certain deadline, the system breaks. In some cases, failure to meet a deadline is unsafe and can even endanger lives. In other cases, missing a deadline does not create safety problems but does create unhappy customers—missed deadlines in printers, for example, can result in scrambled pages.

- *Multirate:* Not only must operations be completed by deadlines, but also many embedded computing systems have several real-time activities going on at the same time. They may simultaneously control some operations that run at slow rates and others that run at high rates.

- Multimedia applications are prime examples of **multirate** behaviour. The audio and video portions of a multimedia stream run at very different rates, but they must remain closely synchronized. Failure to meet a deadline on either the audio or video portions spoils the perception of the entire presentation.

sophisticated functionality

Costs of various sorts are also very important:

- Manufacturing cost: The total cost of building the system is very important in many cases. Manufacturing cost is determined by many factors, including the type of microprocessor used, the amount of memory required, and the types of I/O devices.

- Power and energy: Power consumption directly affects the cost of the hardware, because a larger power supply may be necessary. Energy consumption affects battery life, which is important in many applications, as well as heat consumption, which can be important even in desktop applications.

COVID-19

corona Virus (Covid19) is wreaking havoc in the world. Almost every country is suffering from the Corona Virus. WHO has already announced it a Pandemic disease and many cities are under lockdown situation, people can't step out of their homes, and thousands have lost their lives. Many websites are providing live updates of the coronavirus cases like Microsoft's Tracker, Esri's Covid19 Tracker, etc.

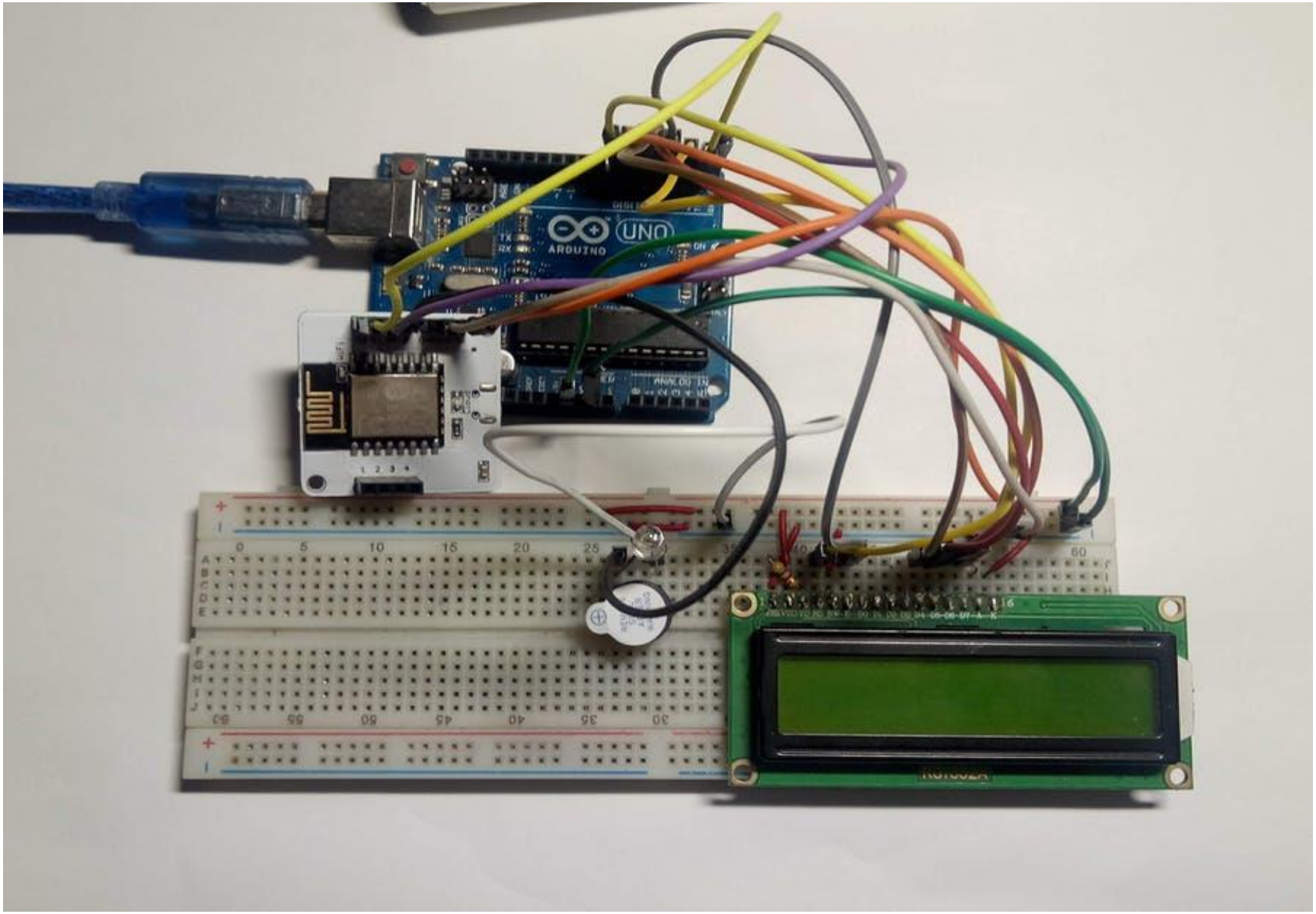
Coronavirus Live Updator

This project does two functions one it's a live updator and trigger the buzzer if the no of cases increases 500 people for like 10min.

Story

This project does two functions. Number one it displays the total number of coronavirus which is as u know its a pandemic.And the second function which its does is to trigger the buzzer when the number off cases around the world have increased by 500 people in the range of 10 minutes to indicate the coronavirus is no joke and to be taken seriously.

The best way to keeping the people alert so that they can be precautions is to constantly reminding them about the no of cases being listed and how dangerous it is.

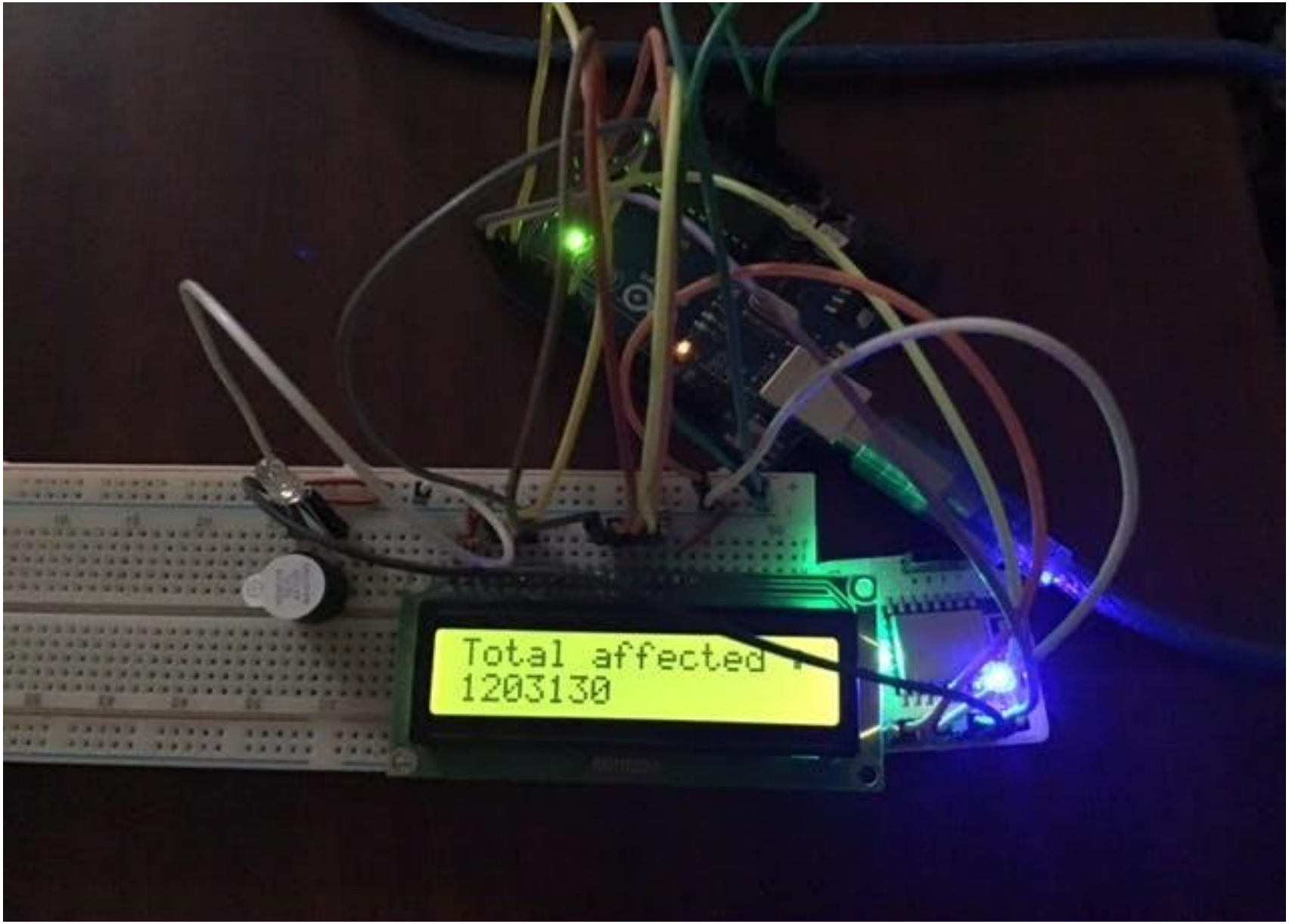

UNO
ARDUINO

Programming in python:

First we get to know about the boltiot library in python u can download it from https://github.com/Inventrom/bolt-api-python and add it to ur libraries in python.

The next thing is to extract data i.e total no of coronavirus Currently registered in the world. The website which I used to extract the data is from https://www.worldometers.info/coronavirus/ I used webscraping in python to extract data. I used urllib library and Beautiful Soup to scarp the website and get the data.

So in python boltiot library we have these functions


Total affected
1203130

Setting up BOLT module:

As we know esp8266 is a very famous module to connect to the internet. So what's bolt then it's the next layer of simplification of esp8266. U can think like arduino is the simplification of how we deal with microcontrollers.

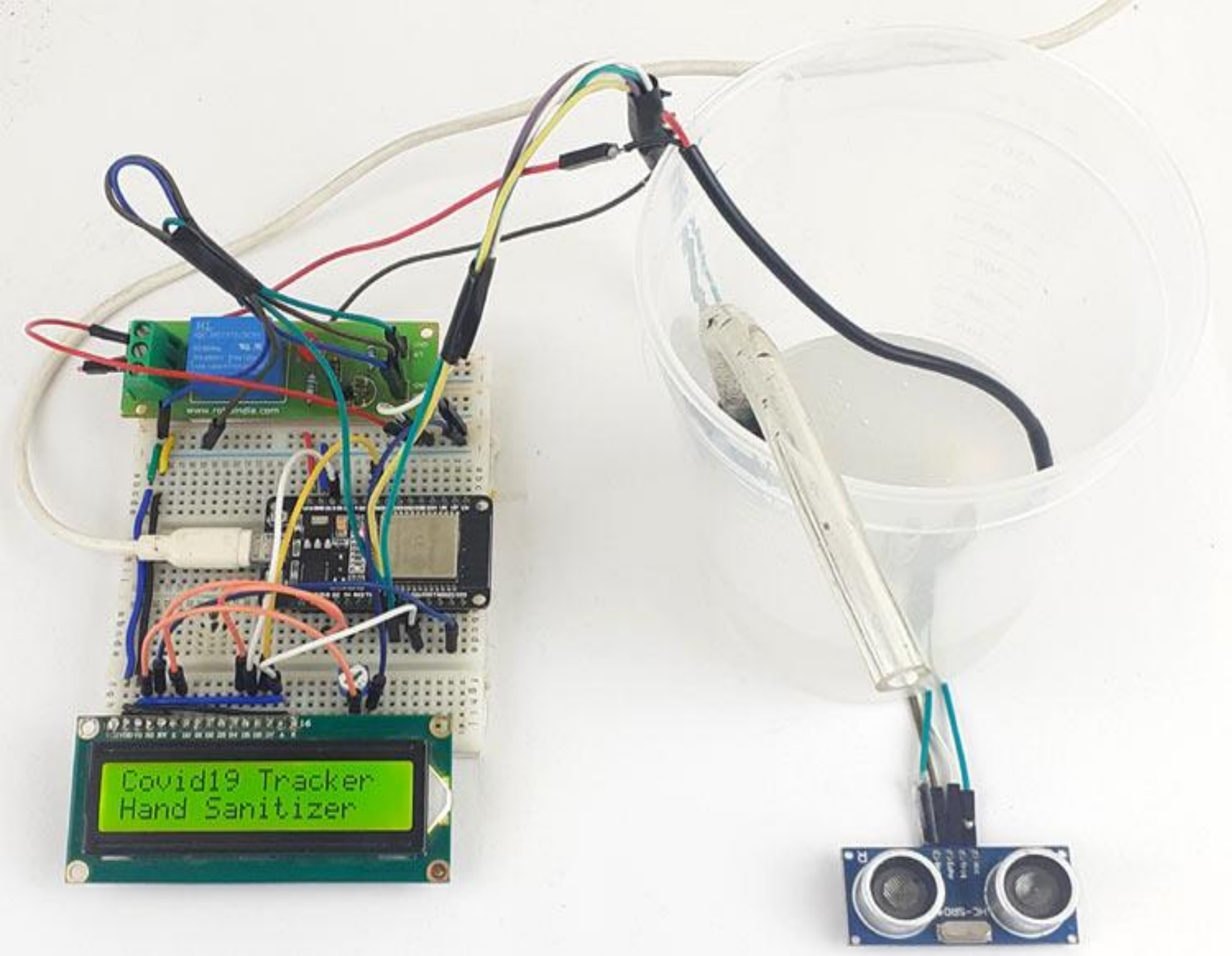
So yaa, bolt has nothing like a fixed language though js and html are fixed ones used but bolt has API(Application Programming Interface).For those who don't know about API google it. So by using API bolt can be controlled like a official language would do.

At first we need to get the API key and DEVICE ID . With these two parameters and boltiot library in python we can control our bolt device. Why DEVICE ID, as there are n number of bolt devices this ID would differentiate from others .

For Controlling bolt u should have a account in www.boltiot.com. After logging into ur account follow the steps.

Automatic Hand Sanitizer Dispenser with COVID19 Live Updates

In this project, we will build an **Auto Hand Sanitizer Dispenser** with an LCD which also shows the live count of Coronavirus cases. This project will use ESP32, Ultrasonic Sensor, 16x2 LCD Module, Water pump, and Hand Sanitizer. We are using Esri's API Explorer to get the live data of Covid19 infected people. An ultrasonic sensor is used to check the presence of hands below the outlet of the sanitizer machine. It will continuously calculate the distance between the sanitizer outlet and itself and tells the ESP to turn on the pump whenever the distance is less than 15cm to push the sanitizer out.
ESP32 is used as the main controller, it is a Wi-Fi module that can easily connect to the internet. We previously used it to build many IoT based projects using ESP32.


Covid19 Tracker
Hand Samitizer

Automatic Hand Sanitizer Dispenser with COVID19 Live Updates

Components Required
ESP32 Dev Module,Ultrasonic Sensor,16*2 LCD Display,Relay Module ,Mini DC Submersible Pump,Hand Sanitizer

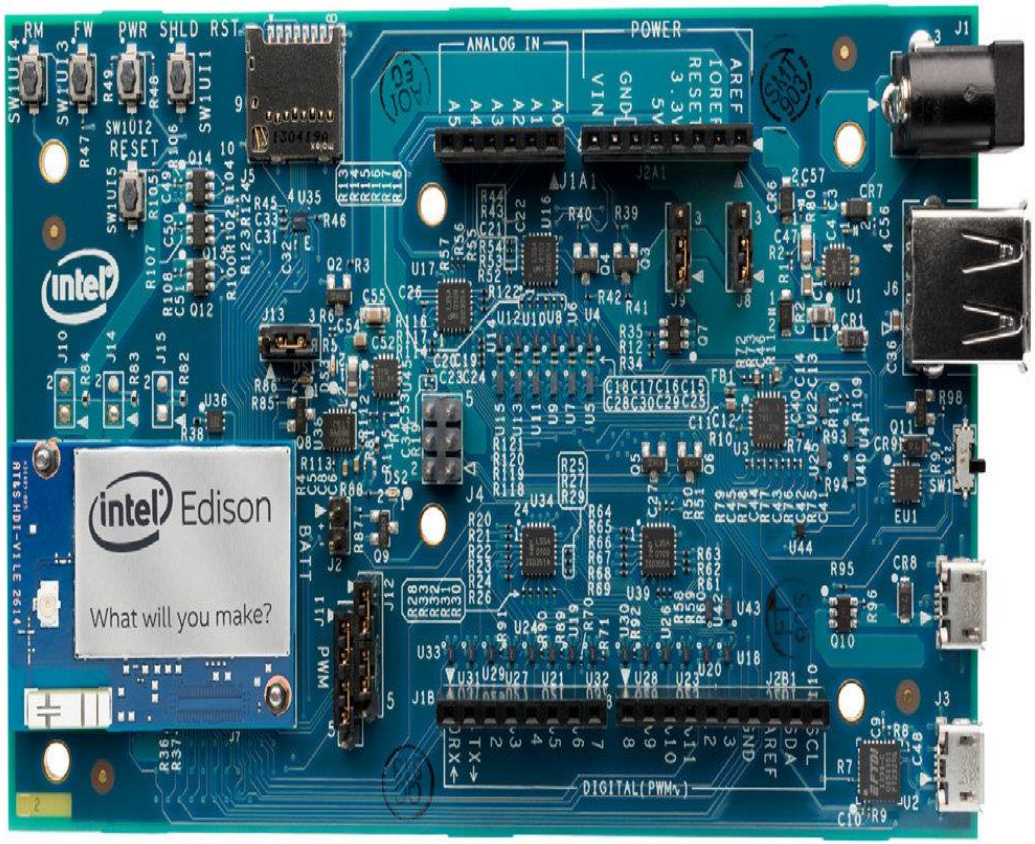
API link for getting the Corona Live Data
Here we need to get the data from the internet and then send it to ESP32 to display it on 16x2 LCD. For that, an HTTP get request is invoked to read the JSON file from the internet. Here we are using the API provided by Coronavirus Disease GIS Hub. You can easily compile the correct query URL to get the total Confirmed and recovered cases for India and can also change the country/Region if you want to use this for a different country.
After getting the JSON data, now generate the code to read the JSON data and phrase it according to our needs. For that, go to the ArduinoJson Assistant and paste the JSON data in the Input section.

Intel® Edison - One Tiny Platform, Endless Possibility

Intel Edison is a tiny Computer made by Intel focusing on IoT and the Wearable Computing.

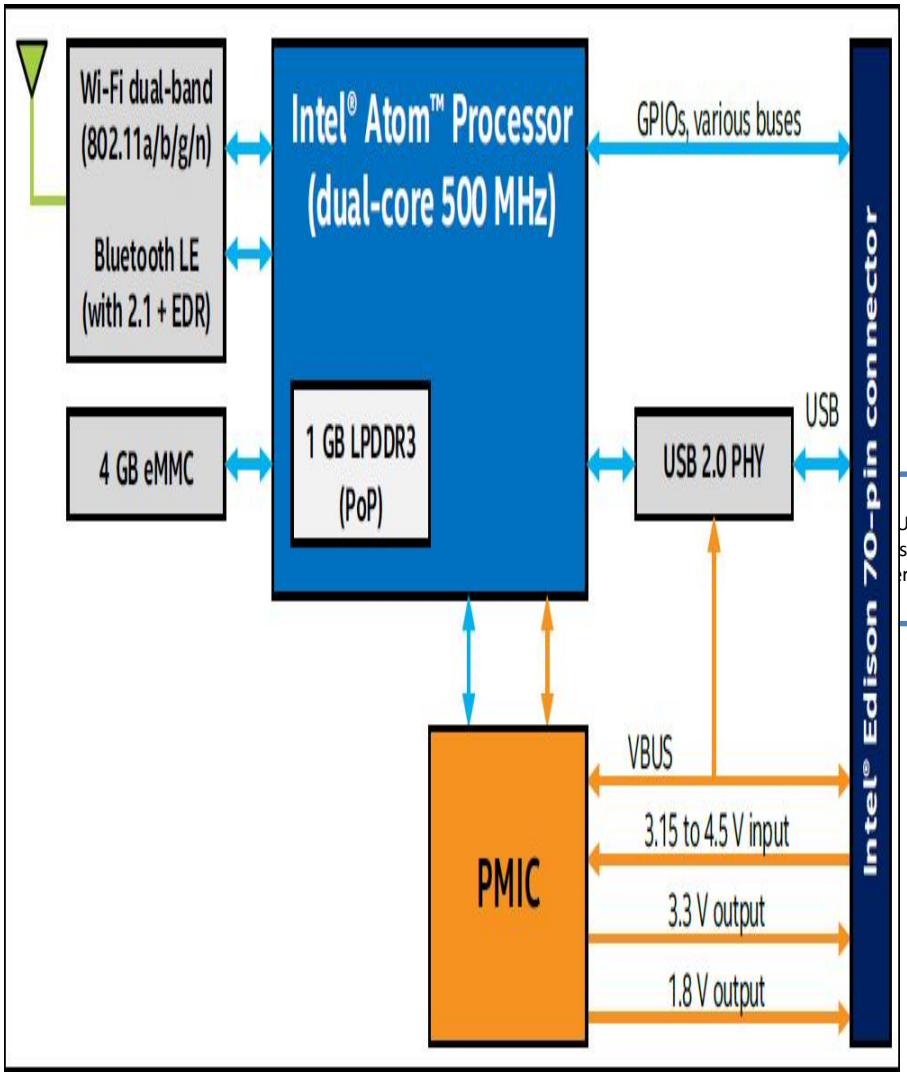
Features:- (Will Amaze You)

- Very Small Form Factor (35.5 × 25.0 × 3.9 mm).
- Inbuilt Dual Band Wi-Fi and Bluetooth (v4.0)
- 500MHz Dual Core Dual Threaded Intel Atom Processor along with 100 MHz Quark Microcontroller
- 1 GB LPDDR3 RAM
- 4 GB Inbuilt Storage
- OTG Support
- Low Power Consumption (Can be powered with single cell Lithium Ion Battery) along with charge controller.
- Etc.

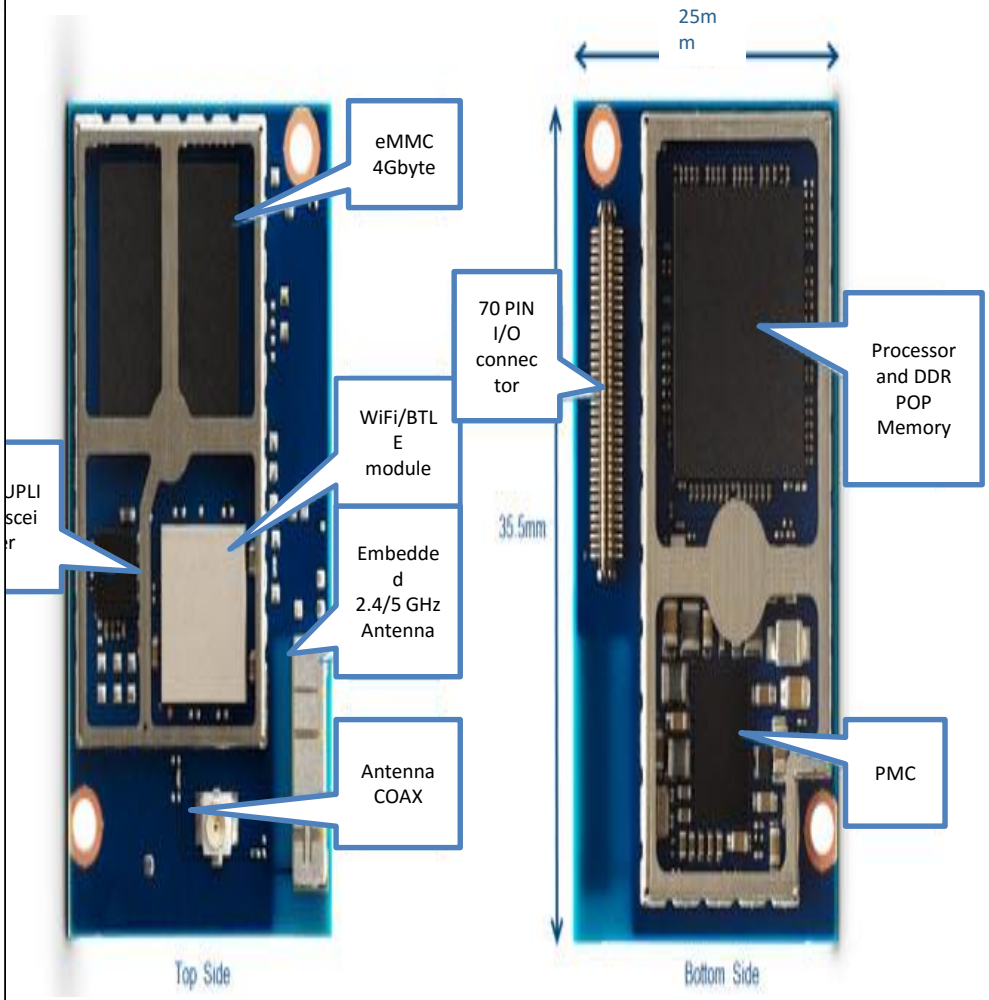


"Tiny computer used as development system for wearable devices for IoT Applications."

Intel® Edison



Intel Edison Block diagram



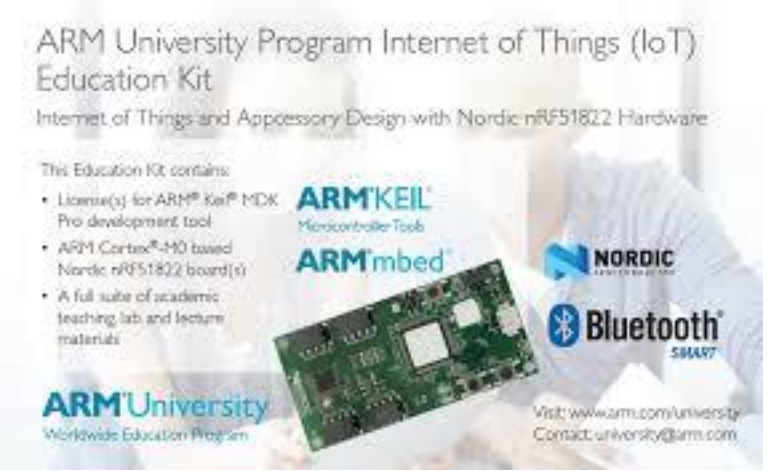
Edison internal detail, front and back

Other IoT Kits/Boards

CC 3200 Launch pad TI

Raspberry Pi

Intel Galileo Gen 2



ARM

Intel® Edison- Application

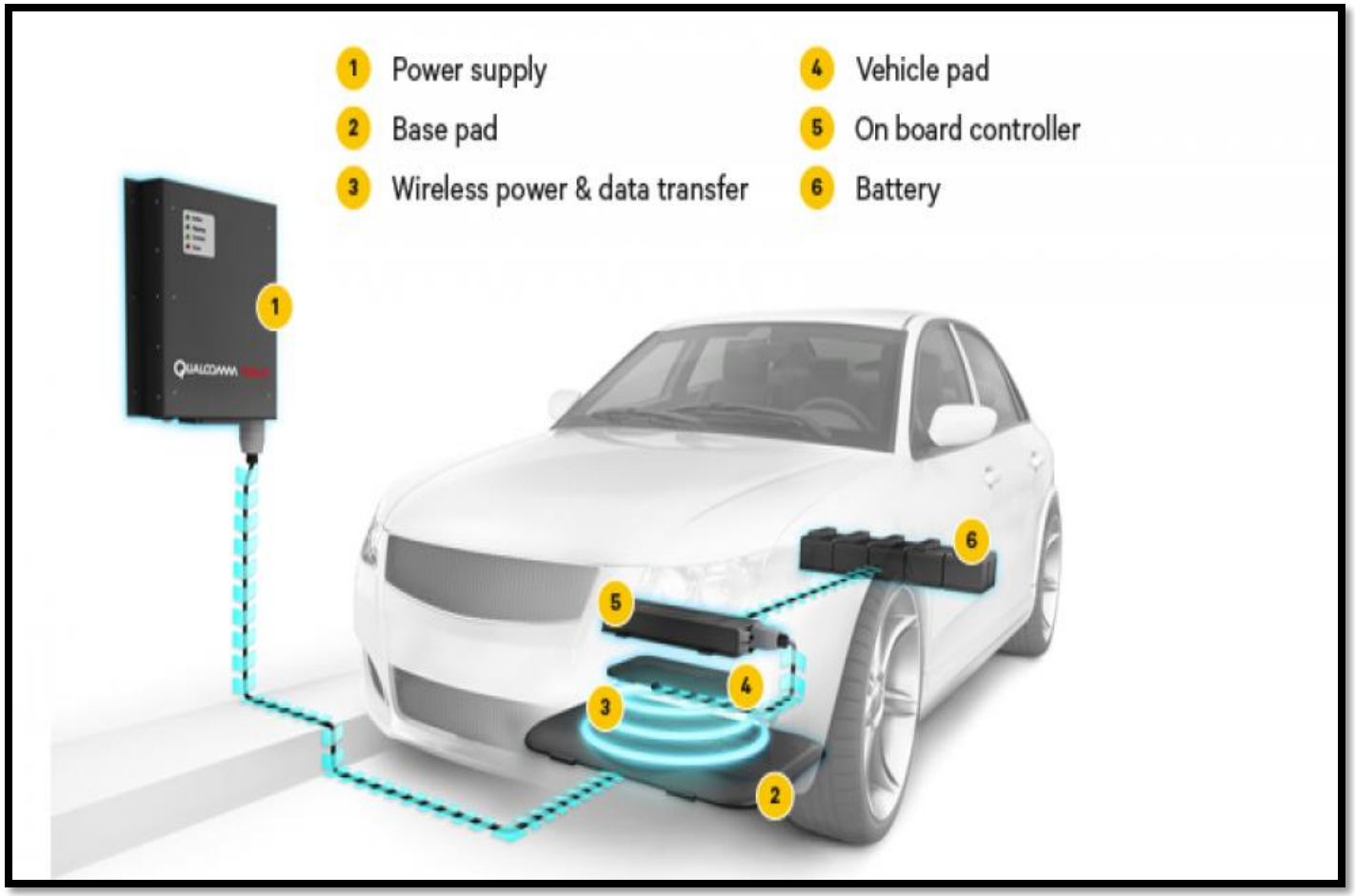
BABYBE
Soft robotics for neonatal healthcare

Qualcomm Helo

Qualcomm halo
Wireless Electric Vehicle Charging

Parked = powering up

Any parking spot fitted with Qualcomm Halo™ technology is a place to recharge your electric car. It's a simple, elegant way to power up, cable-free.



BeagleBoard

The **BeagleBoard** is a low-power open-source single-board computer produced by Texas Instruments in association with Digi-Key and Newark element14. The **BeagleBoard** was also designed with open source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530 system-on-a-chip. **Frequency**: 600 MHz to 1 GHz

Connection: USB On-The-Go

Common manufacturers: Circuitco LLC on beh...

Cost: US$95 to $149

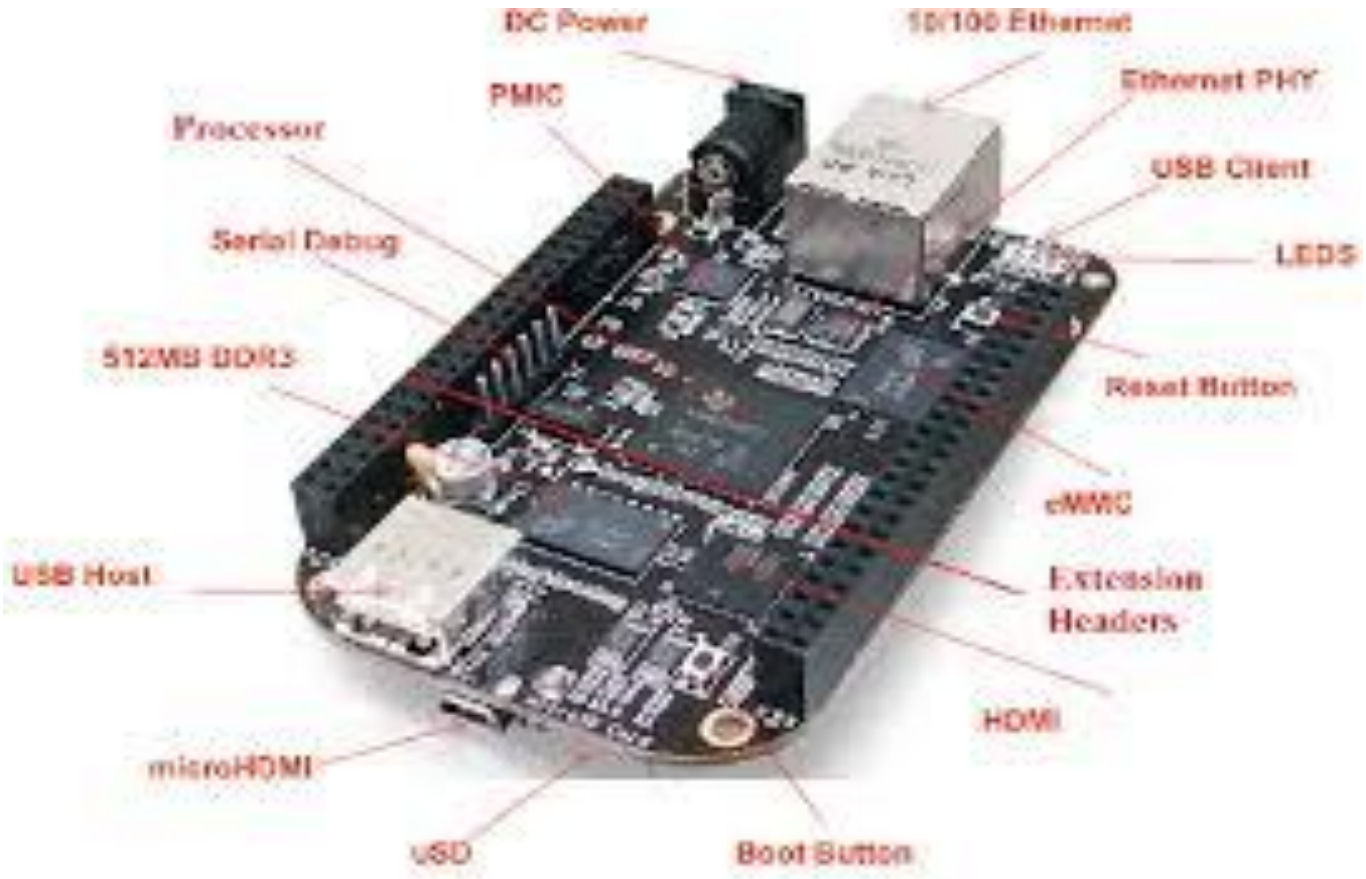
Multiple programming languages: You can write your custom code in almost any language you're most comfortable with: C, C++, **Python**, **Perl**, **Ruby**, **Java**, or even a **shell script**. Linux software: Much of the Linux software that's already out there can be run on the

BeagleBoard

BeagleBoard-X15

BeagleBoard-X15
It is based on the TI Sitara AM5728 processor with two ARM Cortex-A15 cores running at 1.5 GHz, two ARM Cortex-M4 cores running at 212 MHz and two TI C66x DSP cores running at 700 MHz. The used processor provides USB 3.0 support and has a PowerVR Dual Core SGX544 GPU running at 532 MHz.
Processor: ARM Cortex-A8
Introduced: BeagleBoard; July 28, 2008; Beagl...
Ports: USB On-The-Go/DVI-D/PC audio/SDHC...

BeagleBoard



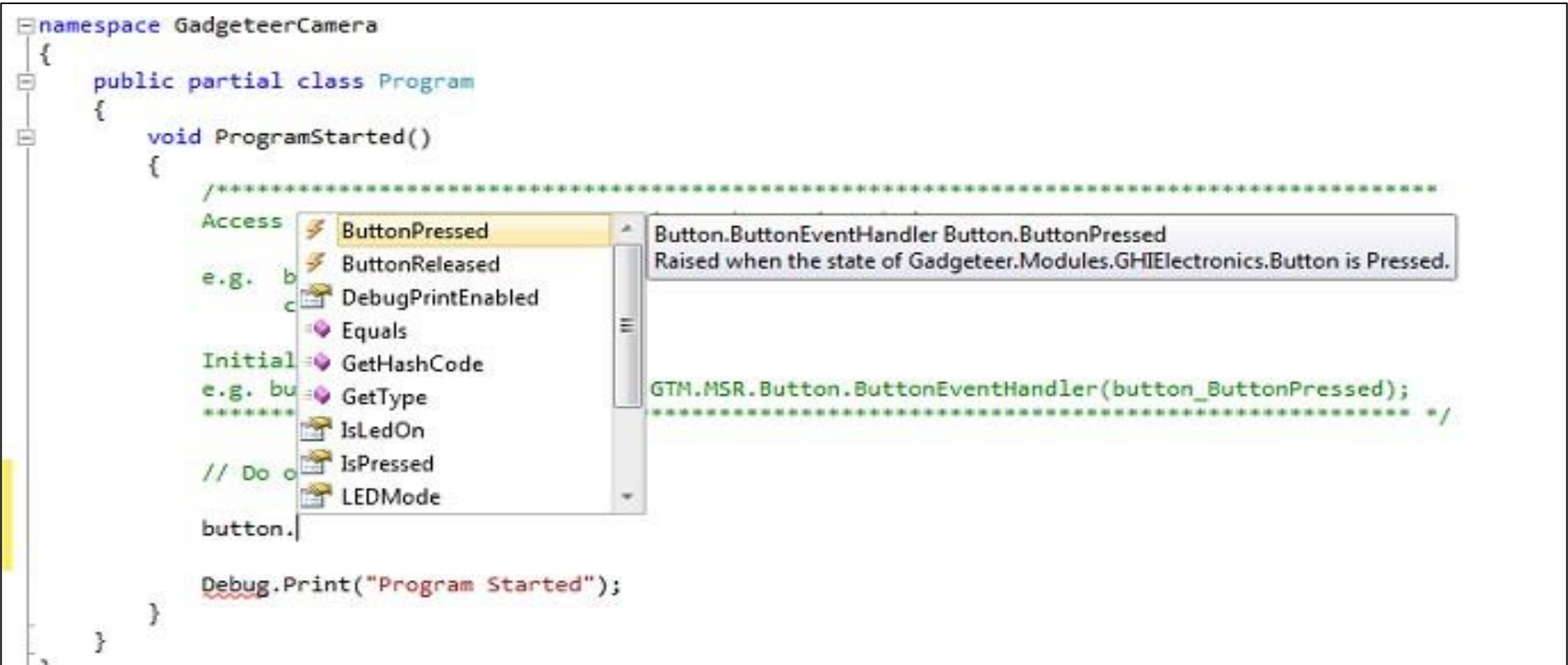
Microsoft .NET Gadgeteer

- **What**
  - fairly new, started in 2011 by Microsoft Research labs
  - An toolkit for building small electronic devices
    - using the .NET Micro Framework and Visual Studio/Visual C# Express.

- **Hardware**
  - a central “mainboard” containing a CPU and several sockets
  - a large number of different modules.

Microsoft .NET Gadgeteer.

- **IDE**
  - Tightly integrated with the Microsoft Visual Studio IDE
    - With features of dynamic syntax checking and continually provides hints and prompts to ease coding.
    - Support debugging via breakpoints, single stepping, variable watches, and execution traces.



Microsoft's Gadgeteer Design Choices

Primary design goal to simplify application development as much as possible

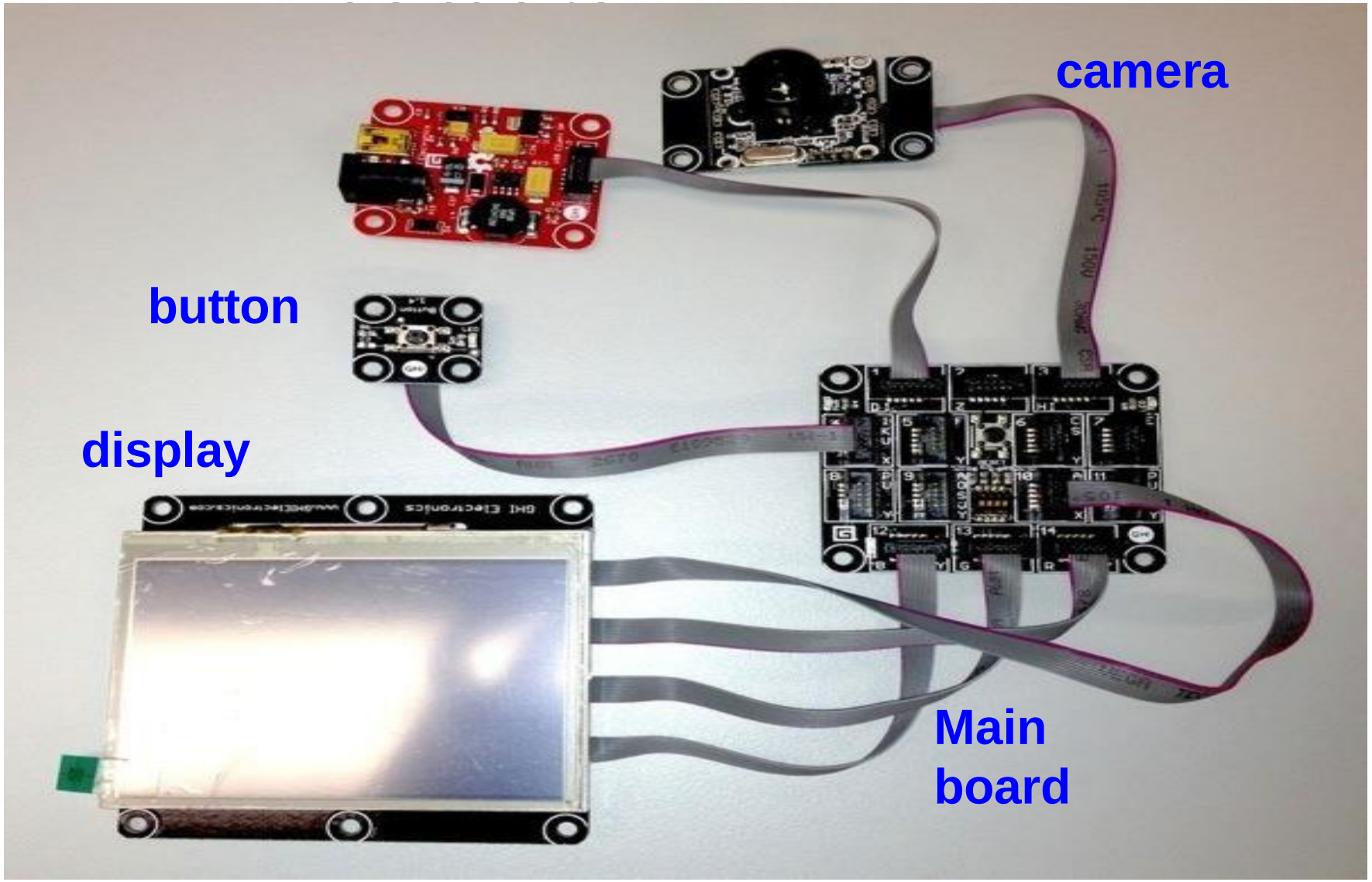
- Prioritized REST-ful support over other Web-related functionality
- Event-based model
  - Simplify the creation of many applications
  - Helps developers familiar with event-based programming on desktop and mobile platforms to transition to embedded device development.
- High-level API
  - Allow modules to be used in sophisticated ways with a few lines of code.

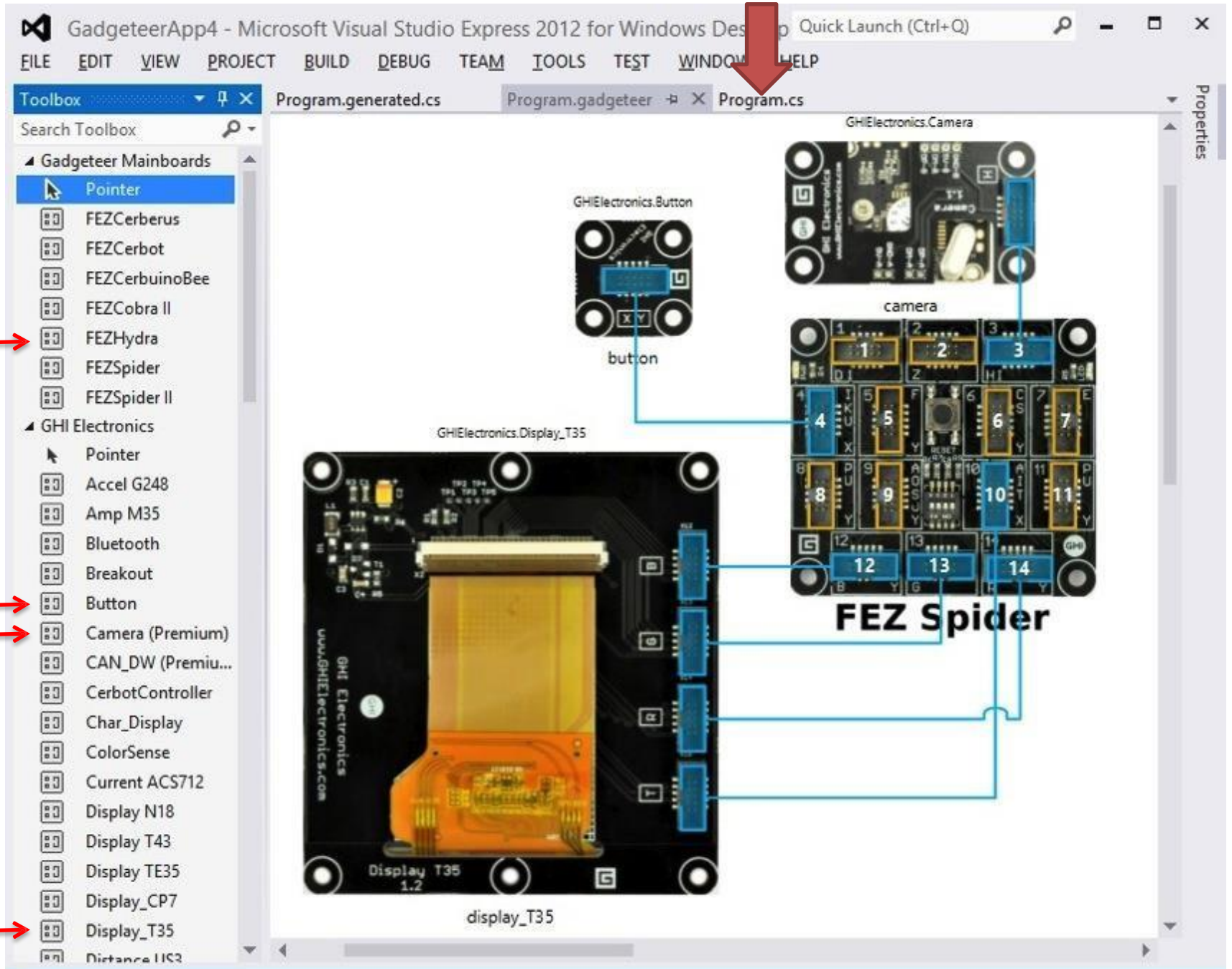
3) Building Web-Connected Devices With Gadgeteer

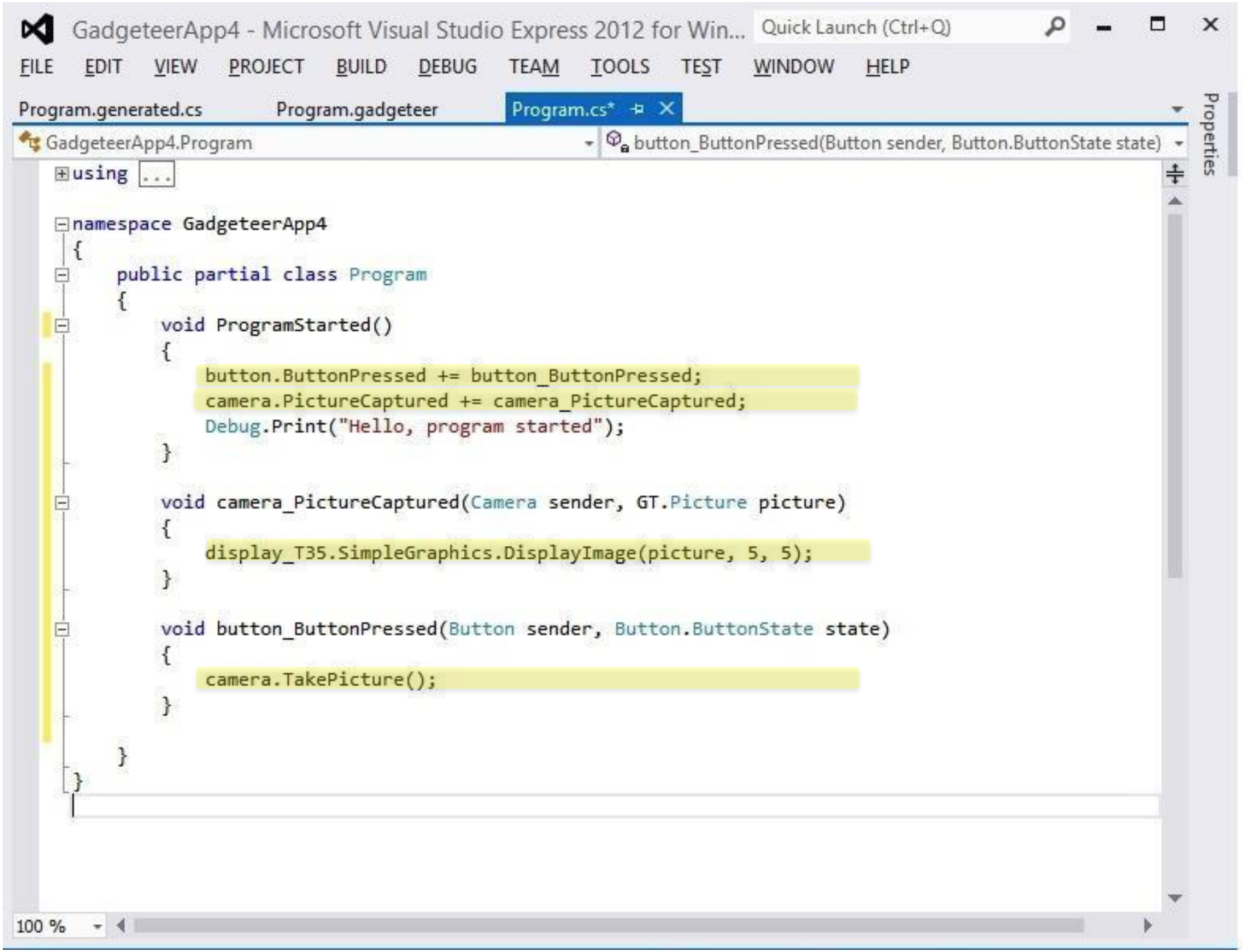
#1. A simple camera
#2. A simple Internet webcam
#3. A sophisticated Web-controlled camera
#4. Logging sensor data using Cloud-based storage
#5. OCR using cloud-based processing

Credit: #2-#5 are shown in:
Steve Hodges, et al. "Prototyping Connected Devices for the Internet of Things." *Computer*, Vol.46, Iss.2, Feb. 2013.

#1. A simple

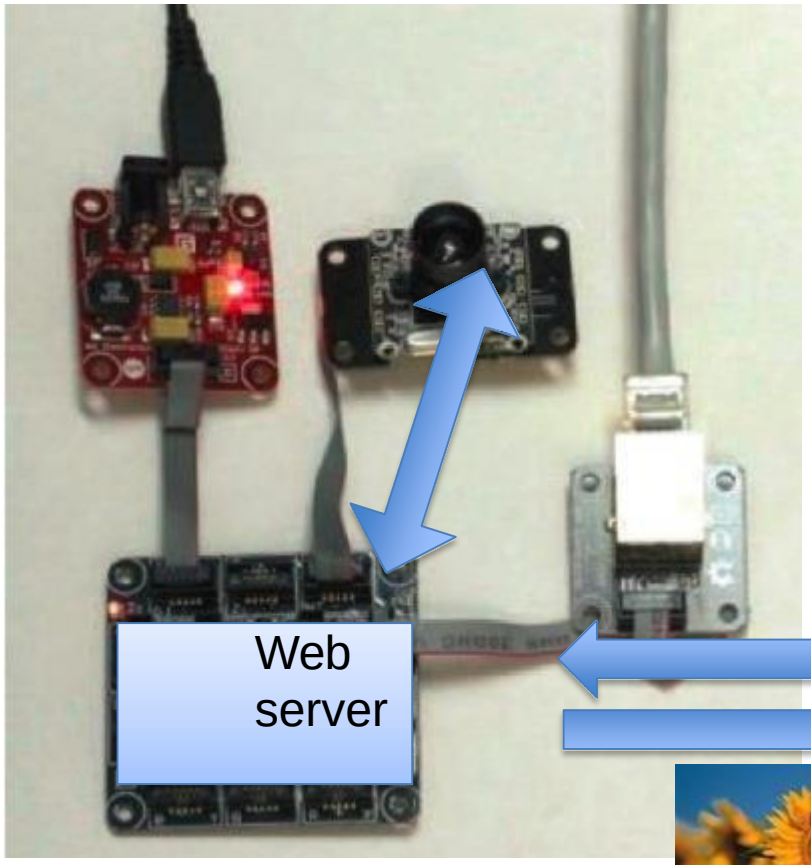



GadgeteerApp4 - Microsoft Visual Studio Express 2012 for Windows Desktop
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST WINDOW HELP
Toolbox
Search Toolbox
Gadgeteer Mainboards
Pointer
FEZCerberus
FEZCerbot
FEZCerbuinoBee
FEZCobra II
FEZHydra
FEZSpider
FEZSpider II
GHI Electronics
Pointer
Accel G248
Amp M35
Bluetooth
Breakout
Button
Camera (Premium)
CAN_DW (Premiu...
CerbotController
Char_Display
ColorSense
Current ACS712
Display N18
Display T43
Display TE35
Display_CP7
Display_T35
Distance US3
Program.generated.cs
Program.gadgeteer
Program.cs
Properties
GHIElectronics.Camera
camera
GHIElectronics.Button
button
GHIElectronics.Display_T35
display_T35
FEZ Spider
Ready

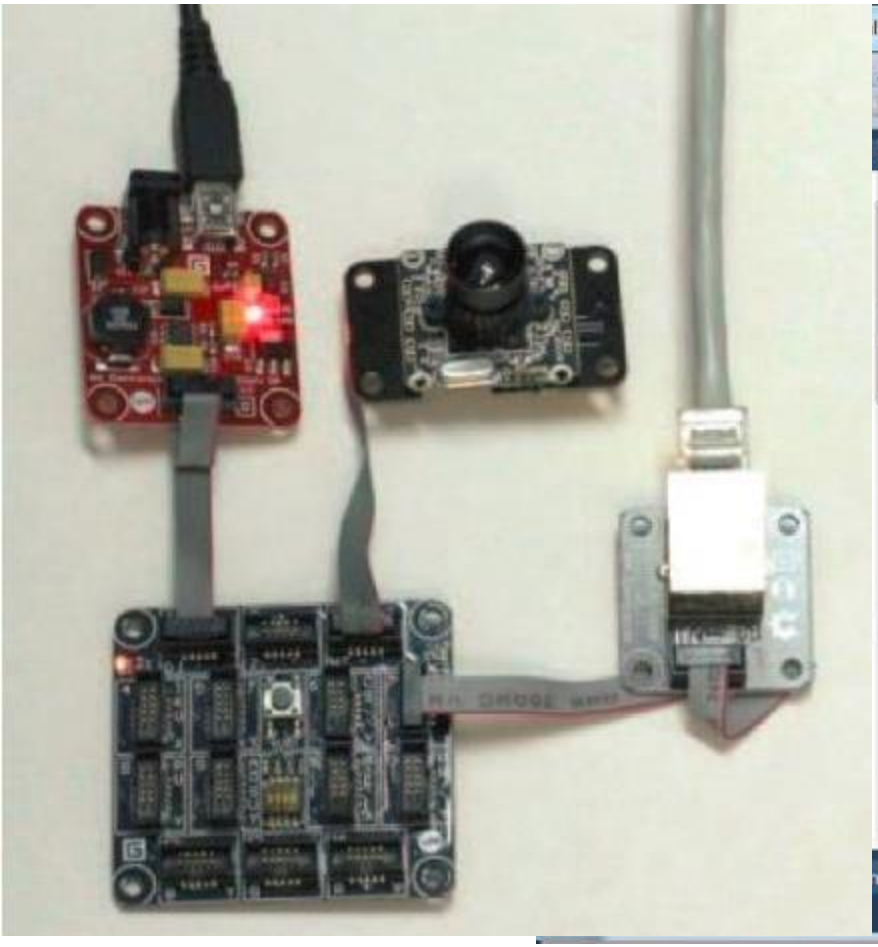

GadgeteerApp4 - Microsoft Visual Studio Express 2012 for Win...
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST WINDOW HELP
Program.generated.cs
Program.gadgeteer
Program.cs*
GadgeteerApp4.Program
button_ButtonPressed(Button sender, Button.ButtonState state)
Properties
using ...
namespace GadgeteerApp4
{
public partial class Program
{
void ProgramStarted()
{
button.ButtonPressed += button_ButtonPressed;
camera.PictureCaptured += camera_PictureCaptured;
Debug.Print("Hello, program started");
}
void camera_PictureCaptured(Camera sender, GT.Picture picture)
{
display_T35.SimpleGraphics.DisplayImage(picture, 5, 5);
}
void button_ButtonPressed(Button sender, Button.ButtonState state)
{
camera.TakePicture();
}
}
}
100 %

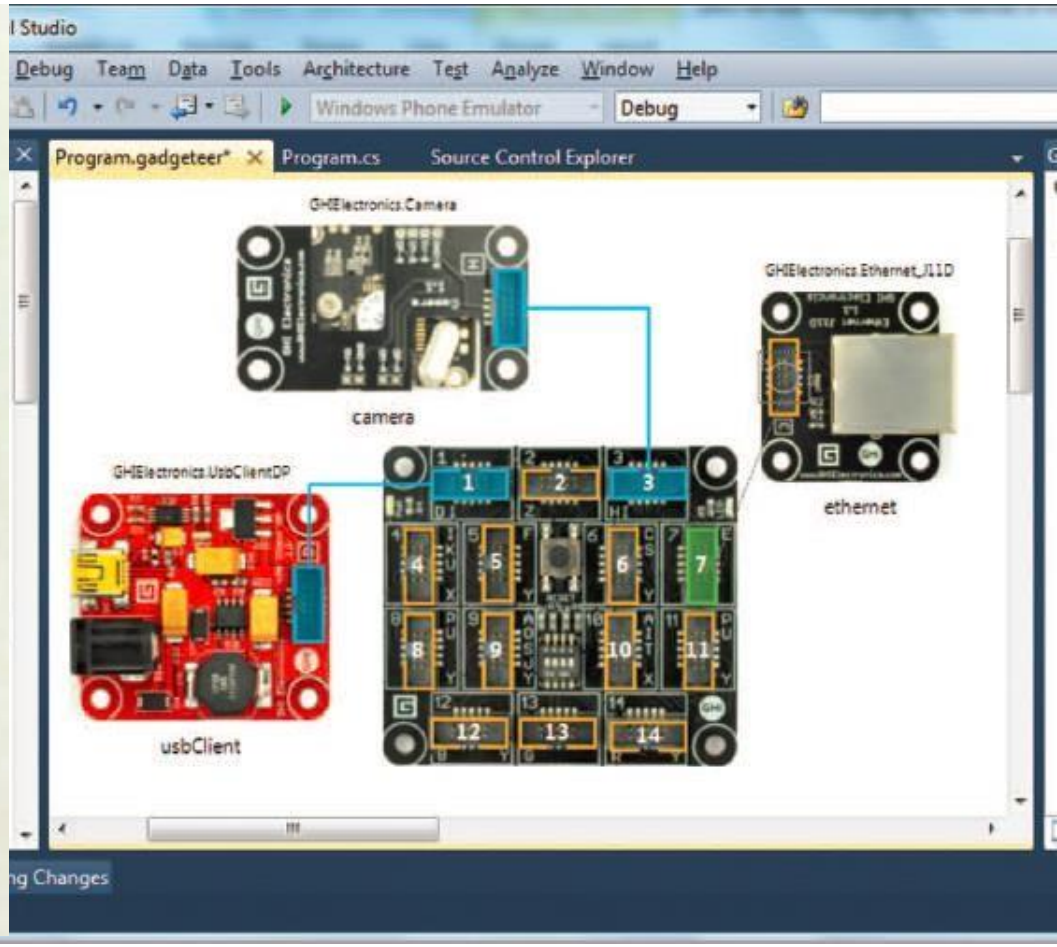
#2. A simple “Internet webcam”

- An HTTP request from a remote client triggers the capture of a new image. The captured image is returned to the Web client.



http://200.200.200.1/picture




Studio
Debug Team Data Tools Architecture Test Analyze Window Help
Windows Phone Emulator
Debug
Program.gadgeteer*
Program.cs
Source Control Explorer
GHIElectronics.Camera
camera
GHIElectronics.Ethernet_J11D
ethernet
GHIElectronics.UsbClientDP
usbClient
1
2
3
4
5
6
7
8
9
10
11
12
13
14
ng Changes


camera


ethernet

```
WebEvent cameraWebEvent;
Responder currentResponder;

void ProgramStarted()
{
    // associate PictureCaptured event with its handler
    camera.PictureCaptured += new Camera.PictureCapturedEventHandler(camera_PictureCaptured);

    // request DHCP address and associate handler for network setup
    ethernet.UseDHCP();
    ethernet.NetworkUp += new NetworkModule.NetworkEventHandler(ethernet_NetworkUp);
}

void ethernet_NetworkUp(GTM.Module.NetworkModule sender,
        GTM.Module.NetworkModule.NetworkState state)
{
    // start a webserver on port 80
    WebServer.StartLocalServer(ethernet.NetworkSettings.IPAddress, 80);

    // set up a handler for http '/picture' requests
    cameraWebEvent = WebServer.SetupWebEvent("picture");
    cameraWebEvent.WebEventReceived += new
      WebEvent.ReceivedWebEventHandler(cameraWebEvent_WebEventReceived);
}

void cameraWebEvent_WebEventReceived(string path, WebServer.HttpMethod method,
        Responder responder)
{
    // initiate a picture and cache the responder to use when the picture is captured
    currentResponder = responder;
    camera.TakePicture();
}

void camera_PictureCaptured(GTM.GHIElectronics.Camera sender, GT.Picture picture)

{
    // respond to web request with the picture
    currentResponder.Respond(picture);
}
```

4

1

3

2

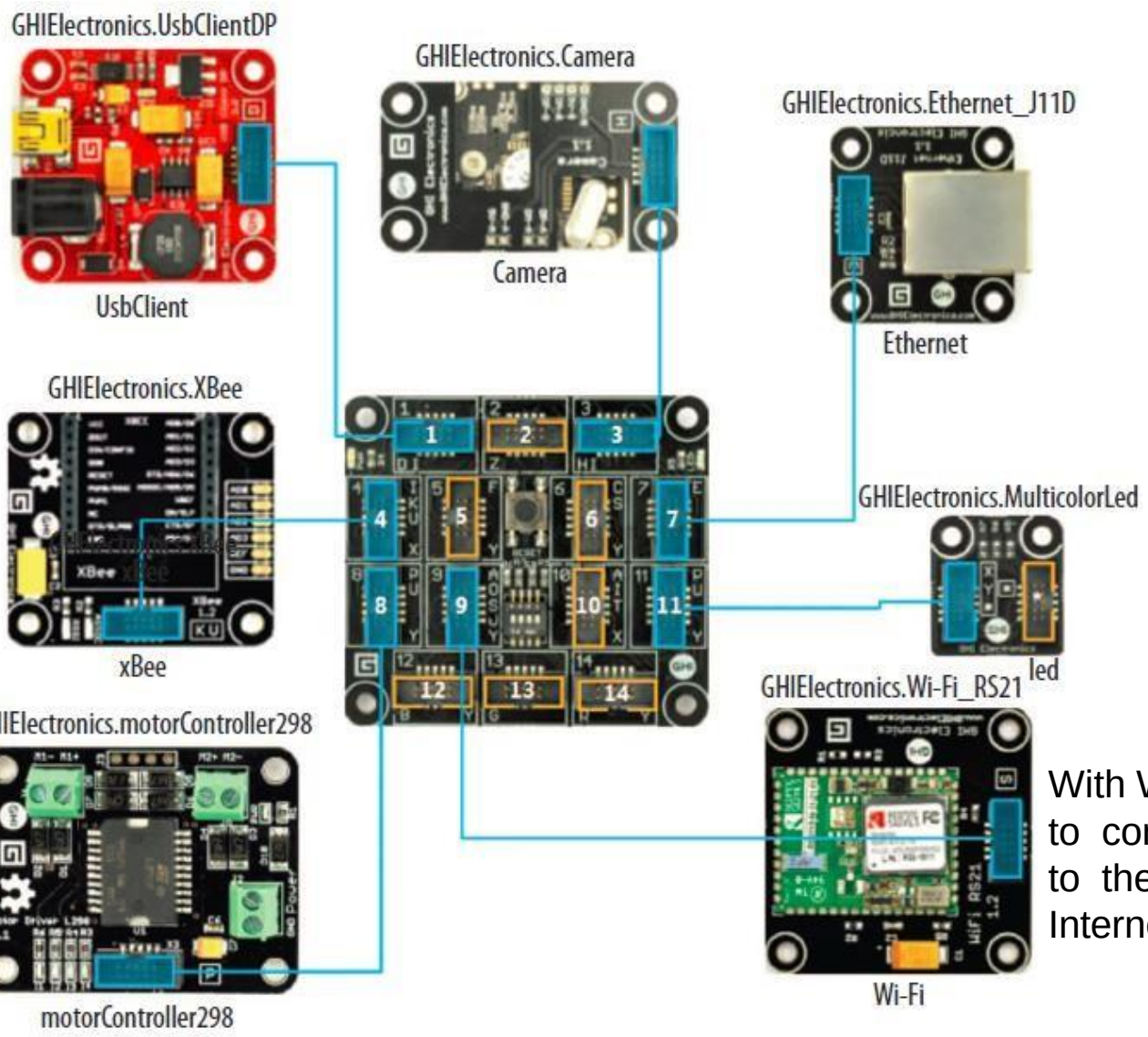
3

4

#3 A more sophisticated Web-controlled camera

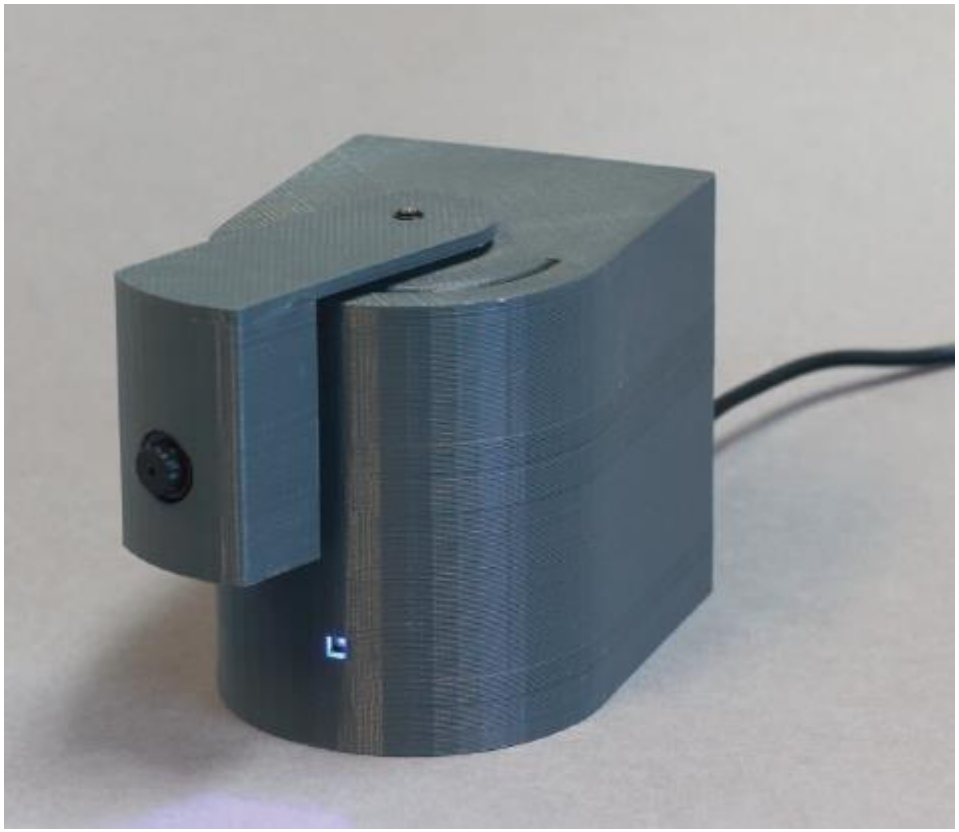
With Zigbee to connect to ligher-weight Gadgeteer devices e.g., temperature sensor.

With a servomotor-controlled arm, allowing remote panning as well as image capture, again over a REST-ful interface.



With Wi-Fi to connect to the Internet

#3 A more sophisticated Web-controlled camera



With a 3D-printed plastic enclosure

#4 Logging sensor data using Cloud-based storage

Senso
r

IoT
(Web
client)

Create a feed (POST) Updating a feed (PUT) Listing all feeds (GET) Viewing a feed (GET)

HTTP

REST-ful Web
server

- A key benefit of connected operation is the potential to leverage cloud-based computation
- The Gadgeteer libraries were designed to ensure that making a Web request is as straightforward as receiving one

#4. Cloud-based processing for connected devices

Sensor

IoT
(Web client)

Create a feed (POST) Updating a feed (PUT) Listing all feeds (GET) Viewing a feed (GET)

HTTP

REST-ful Web server

```
PUT /v2/feeds/105259.csv HTTP/1.1
User-Agent: My_Project
Host: api.cosm.com
X-ApiKey: 862379f7858ed028ba53cd708c6bdcef7b8beb75d7704be96efbc521696d014
Content-Length: 11
Content-Type: txt/csv
Connection: close

test, 200
```

1 *Feed ID*

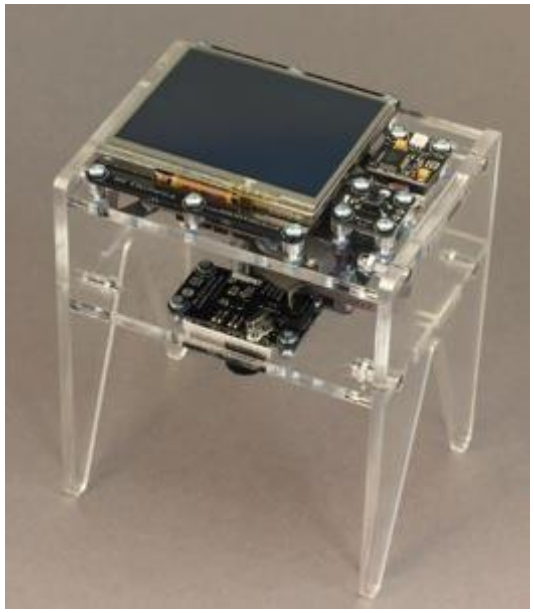
2 *API key*

3

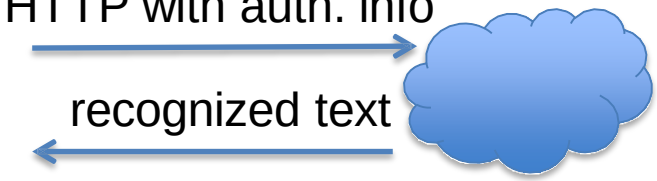
4 *data name & data value*

```
HttpRequest request = HttpHelper.CreateHttpPutRequest("http://api.pachube.com/v2/fees/105259.csv",
          PUTContent.CreateTextBasedContent("test,"+sensorDate.Temperature.ToString()"), "text/csv");

request.AddHeaderField("X-ApiKey", 862379f7858ed028ba53cd708c6bdcef7b8beb75d7704be96efbc521696d014;

request.ResponseReceoved += new Httprequest.ResponseHandler(req_ResponseReceived);

request.SendRequest();
```

#5 OCR using cloud-based processing



- When the shutter button is pressed, it sends the image to the Project Hawaii service for OCR processing, and displays the returned text on its LCD screen




```
void ProgramStarted()
{
    ethernet.UseDHCP();
    button.ButtonPressed += new Button.ButtonEventHandler(button_ButtonPressed);
    camera.PictureCaptured += new Camera.PictureCapturedEventHandler(camera_PictureCaptured);
}

void button_ButtonPressed(Button sender, Button.ButtonState state)
{
    camera.TakePicture();
}

void camera_PictureCaptured(Camera sender, GT.Picture picture)
{
    // Show the picture on the display
    display.SimpleGraphics.DisplayImage(picture.MakeBitmap(), 0, 0);

    // create and send an HTTP request which will send the picture to the Hawaii OCR service
    HttpRequest request = HttpHelper.CreateHttpPostRequest("http://157.55.188.73/OCR",
       POSTContent.CreateBinaryBasedContent(picture.PictureData), "image/jpeg");
    request.AddHeaderField("Authorization", "Basic " +
       ConvertBase64.ToBase64String(Encoding.UTF8.GetBytes("<insert your appID here>"));
    request.AddHeaderField("Cache-Control", "no-cache");
    request.ResponseReceived += new HttpRequest.ResponseHandler(request_ResponseReceived);
    request.SendRequest();
}

void request_ResponseReceived(HttpRequest sender, HttpResponse response)
{
    // for this example we just display the first OCR'ed word returned by Hawaii
    // by looking between the "<Text>" and "</Text>" tags
    int start = response.Text.IndexOf("<Text>", 0) + 6;
    int end = response.Text.IndexOf("</Text>", 0);
    display.SimpleGraphics.DisplayText(response.Text.Substring(start, end - start),
        Resources.GetFont(Resources.FontResources.NinaB), GT.Color.Red, 0, 0);
}
```

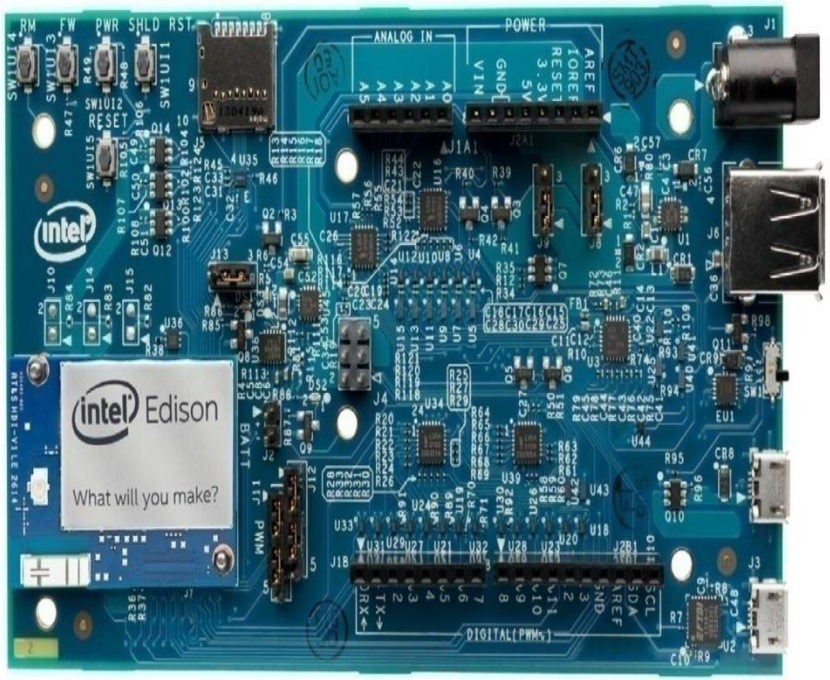
4) Comparison and Recommendation

	Arduino	Gadgeteer	Raspberry Pi
Model	R3	FEZ Spider	Model B
Price	$30	$120	$35
Processor	ATMega 328	ARM7	ARM11
Clock Speed	16MHz	72MHz (168MHz for FEZ Cerberus)	700MHz
RAM	2KB	16MB	256MB
Flash	32KB	4.5MB	(SD Card)
Min Power	42mA (0.3W)	160mA	700mA (3.5W)
Dev IDE	Arduino Tool	Visual Studio	IDEL, Scratch, Squeak/Linux
	Low-level C++	Managed code Concise code Real-time debugging Excellent modular concept, very good for code and hardware re-use.	being programmed in many different languages
	run one program at a time	run one program at a time	capable of running multiple programs at the same time
			Ethernet, 2USB HDMI, Composite

Case Study → Plant Monitor: Agro-Electronics

If you have done some plantation in a remote area or may be if you are a farmer then you might have an idea then you can understand how difficult it is to monitor proper moisture.

So our team Decided to come up with a solution for this problem using IoT.
For this we used the Intel Edison.



Intel Edison with Arduino Breakout Board

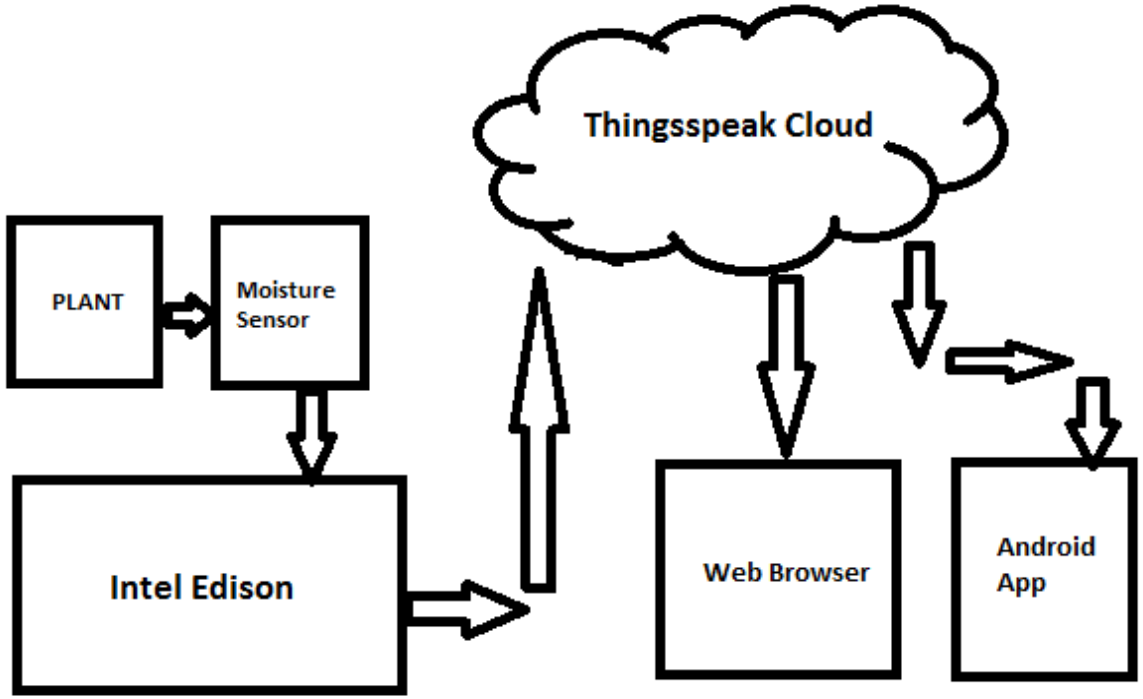
Smart Vending Machine

Whether Monitoring System

Case Study → Plant Monitor

Er. Ishwar Rathod

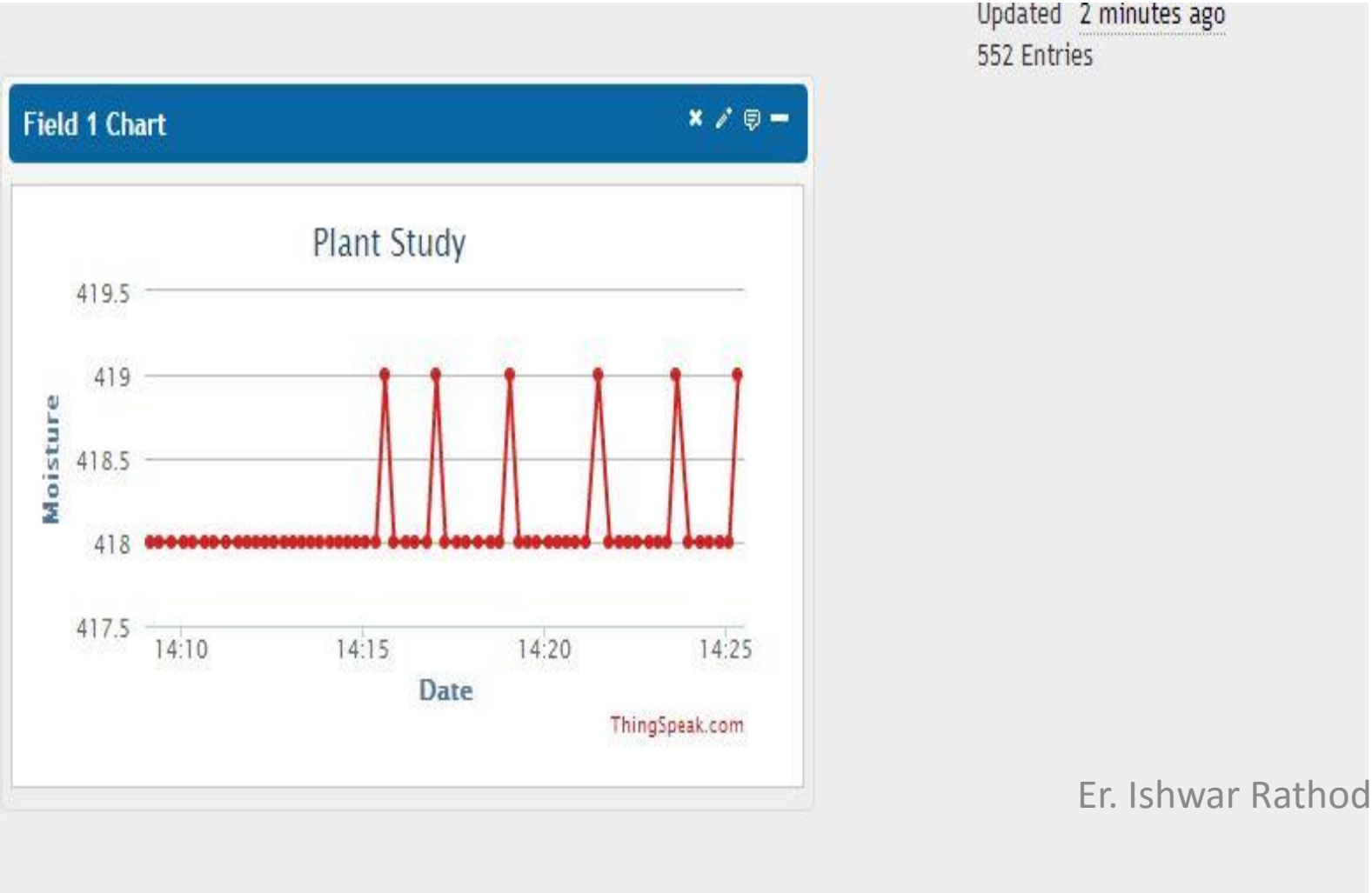
Hardware Block



Case Study → Plant Monitor

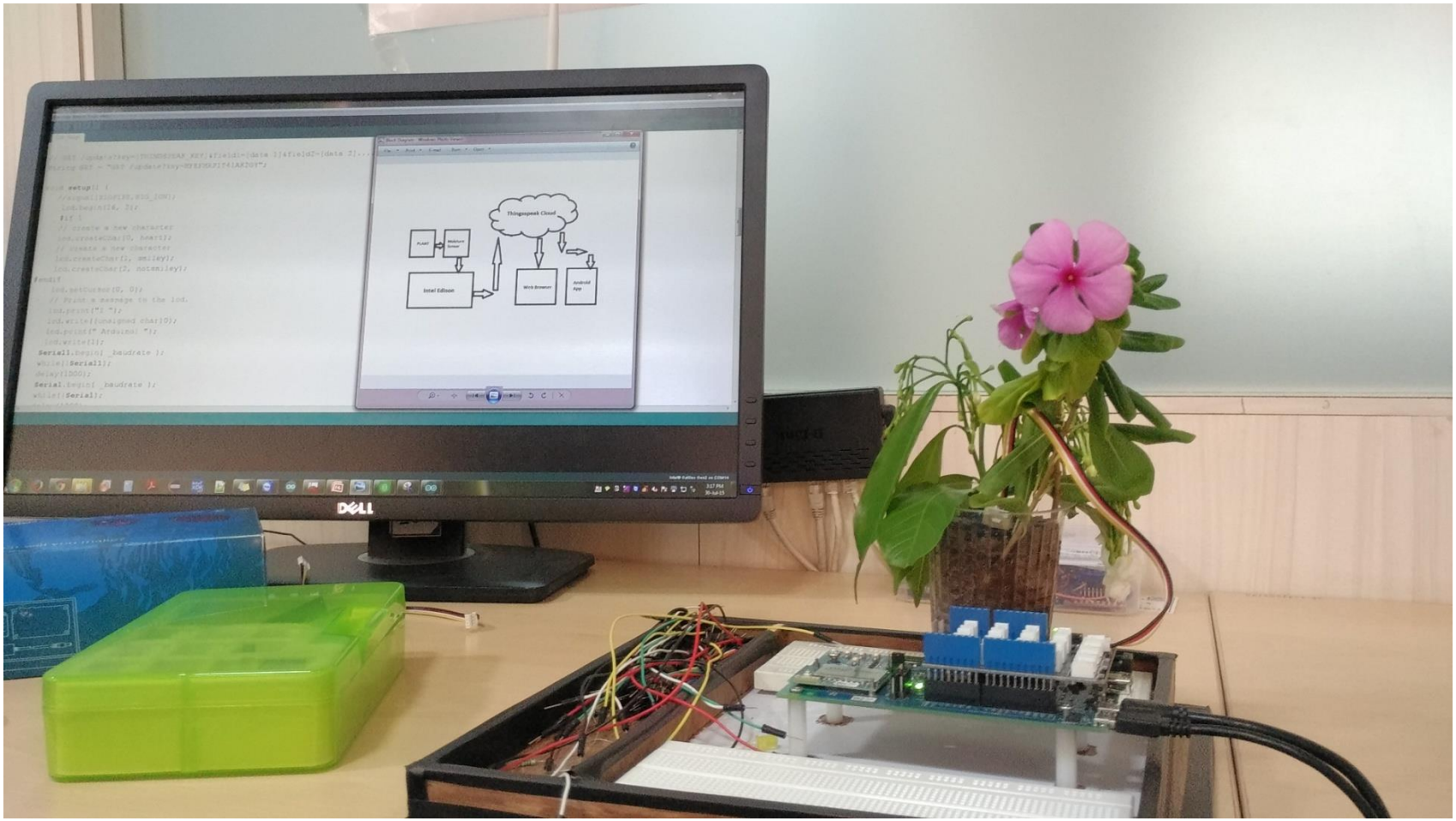
Monitoring Block

Things speak
Channel Data



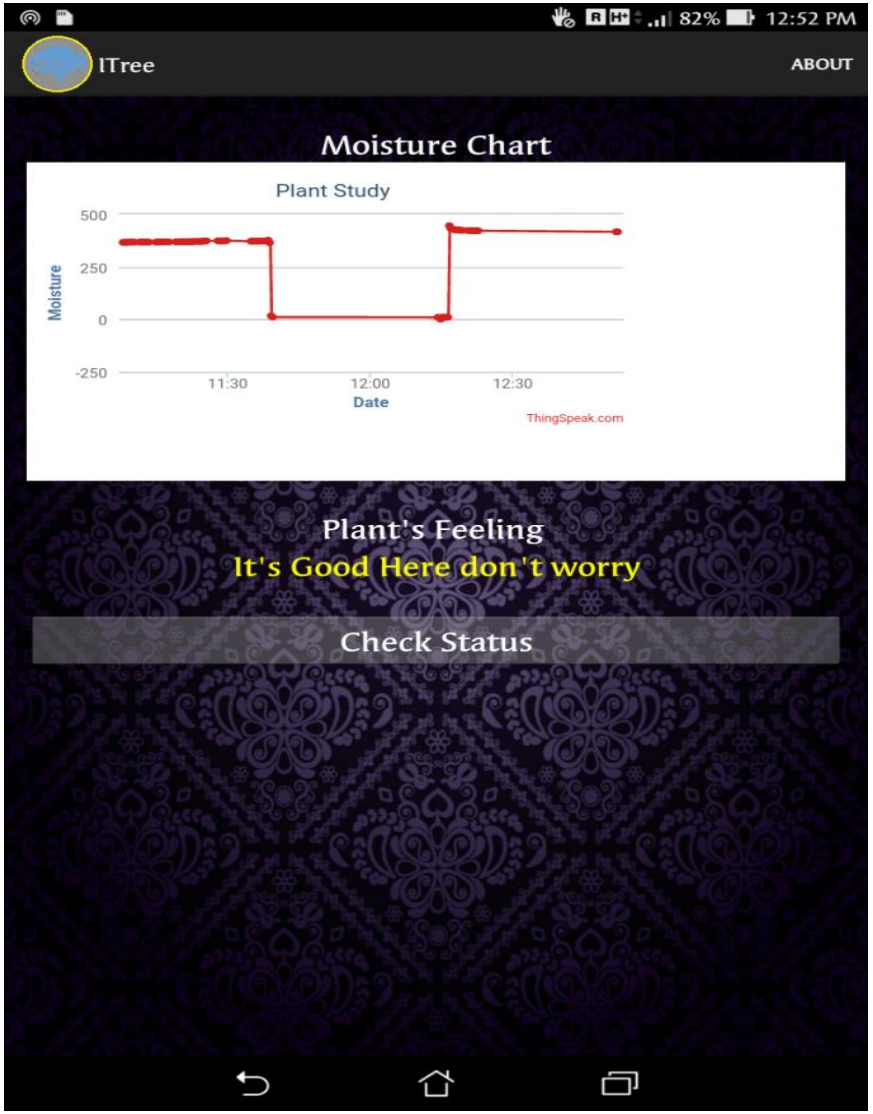
Er. Ishwar Rathod

Case Study → Plant Monitor

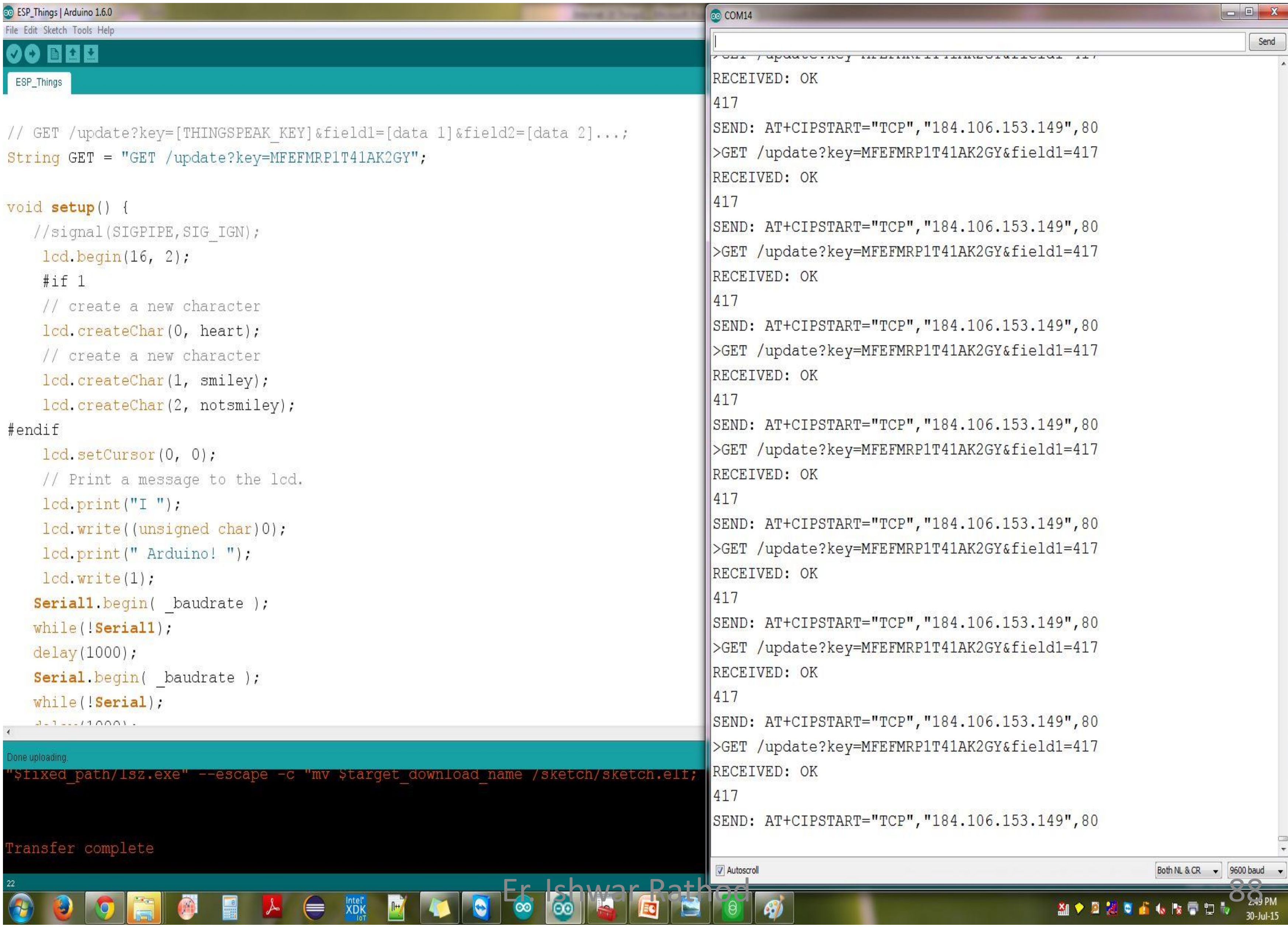


Android App


82% 12:52 PM
ITree
I Tree
Plant Monitor


82% 12:52 PM
ITree
ABOUT
Moisture Chart
Plant Study
500
250
0
-250
Moisture
11:30
12:00
12:30
Date
ThingSpeak.com
Plant's Feeling
It's Good Here don't worry
Check Status

Programming



Future, Challenges and Opportunities

Internet of Things is a truly interdisciplinary field.

Big Firms like Forbes and Gartner have already predicted that there will be at least 75 billion devices connected to Internet by 2020.

The Area of IoT is in its very beginning yet its getting such a huge attention, so it can be easily said that in future we are going to see a lot of Internet of Things.

Security, Trust and Privacy, Integration, Architecture and Protocols etc.

Startups, Entrepreneur, Enterprises

Recommendatns

- **For applications minimizing size -> Arduino**
  - There are very small Arduino embedded systems for making very tiny little gadget.
- **For battery powered applications -> Arduino**
  - Uses the least power of the bunch.
  - Work with a wide range of input voltages (so support a variety of different types of batteries).
- **For applications that interface to external sensors -> Arduino, Gadgeteer**
  - Arduino: lots of external (cheap) sensors
  - Gadgeteer: many slots available for connecting multiple sensors easily.
- **For applications that connect to the internet -> Gadgeteer, Raspberry Pi**
  - Gadgeteer: Full TCP/IP Stack with SSL, HTTP, TCP, UDP, DHCP
  - Raspberry Pi: the Linux OS has many components built-in that provide rather advanced networking capabilities.
- **For applications that use a graphical user interface -> Raspberry Pi**
  - It has an HDMI output.
  - A fully functional computer with graphical user interface.

Production

- No matter which tools are used for prototyping, when large-scale deployments or mass production is needed, it is more cost- effective to move to a custom PCB
  - as it can be made more cheaply and compactly through circuit integration.

Conclusions

- Different tools may be suitable for different kinds users, e.g.,
  - developers,
  - researchers,
  - designers,
  - educator, and
  - hobbyists.
- As the IoT vision gradually becomes a reality, using connected-device prototypes to explore the design space will be important.

Thank you all !