SKETCHING
FAQ

1. What is sketching? Explain its role in prototyping.
2. Write a note on sketching.

- The first step you'll take when working on your prototype will be to jot down some ideas or draw out somedesign ideas with pen and paper.
- That is an important first step in exploring the idea and one we'd like to extend beyond the strict definition toalso include sketching in hardware and software.
- It means the process of exploring the problem space: iterating through different approaches and ideas to workout what works and what doesn't.
- The focus isn't on fidelity of the prototype but rather on the ease and speed with which you can try things out.

Role of sketching in prototyping

- While sketching on paper helps designers to give a form to their ideas, sketching in hardware takes it one stepfurther and allows them to explore possibilities and feasibility of their designs in a more physical way.
- It is an archetypal activity associated with design.
- Sketches are:
  - **Quick:** A sketch is quick to make or at least gives an impression that it is so.
  - **Timely:** A sketch can be provided whenever needed
  - **Inexpensive:** A sketch is cheap. Cost must not inhibit (restrict) the ability to explore a concept, especially inthe early design process.
  - **Disposable:** Sketches are disposable. The investment with a sketch is in the concept not execution
  - **Plentiful:** Their meaning is in the context of a collection or series, not as an isolated rendering
  - **Clear Vocabulary:** A sketch is rendered always with certain conventions that it follows.
  - **Ambiguity:** They are intentionally ambiguous. I.e. they can be interpreted in many ways.
  - **Suggest and explore rather than confirm:** Suggestions come upon discussions to create a better prototype.

FAMILIARITY

- Familiarity is the option of using a familiar or comfortable language to prototype an IoT Project.
- For example, if one can already program like a whiz in Python, maybe picking a platform such as Raspberry Pi, which lets the person write the code in a language that he already knows would be a better option rather thanhaving to learn Arduino from scratch.

COSTS VERSUS EASE OF PROTOTYPING
FAQ

1. How can we decide between the cost and ease of prototyping?
2. Discuss the tradeoffs between Costs versus ease of prototyping.
3. What factors should be considered when deciding between the cost and ease of prototyping?

(Tradeoff means exchanging something of value, especially as part of a compromise.)

- The relationship between the costs (of prototyping and mass producing) of a platform against the developmenteffort that the platform demands must always be considered.
- This trade-off is not hard and fast, but it is beneficial if you can choose a prototyping platform in a performance/capabilities bracket similar to a final production solution.

- In that way, one will be less likely to encounter any surprises over the cost, or even the wholesale viability ofyour project, down the line.
- For example, there are many ways we can develop an electronic device. They are:

Using a microcontroller

- The cheapest possible way of creating an electronic device might currently be an AVR microcontroller chip, which can be purchased from a component supplier for about £3.
- This amount is just for the chip, so one should understand the details of how to connect the pins to other components and how to flash the chip with new code.
- For many people, this platform would not be viable for an initial prototype.

Using a simple microcontroller based development board

- Stepping upwards to the approximately £20 mark, one could look at an Arduino or similar.
- It would have exactly the same chip, but it would be laid out on a board with labeled headers to help you wireup components more easily, have a USB port where you could plug in a computer, and have a well-supported IDE to help make programming easier.

Using a single board computer

- For more money again, approximately £30, you could look at the Beagle Bone, which runs Linux and has enough processing power and RAM to be able to run a high-level programming language: libraries are provided within the concurrent programming toolkit Node.js for JavaScript to manipulate the input/output pins of the board.

PROTOTYPE AND PRODUCTION

FAQ

1. What are the challenges when we move from prototype to mass production? Explain.
2. Explain the transition from prototyping to production.
3. Describe the difficulties encountered during the transition from a prototype to mass production.

Changing embedded platform

- When you scale up, you may well have to think about moving to a different platform, for cost or size reasons.
- If you’ve started with a free-form, powerful programming platform, you may find that porting the code to a more restricted, cheaper, and smaller device will bring many challenges.
- This issue is something to be aware of.
- If the first prototype you built on a PC, iPhone, Beagle Bone, or whatever has helped you get investment or collaborators, you may be well placed to go about replicating that compelling functionality on your final target.

Physical prototypes and mass personalization

- Chances are that the production techniques that you use for the physical side of your device won’t translate directly to mass production.
- Usually 3D printing is used for the physical prototypes.
- However, while the technique might change—injection molding in place of 3D printing, for example—in mostcases, it won’t change what is possible.

Climbing into the cloud

- The server software is the easiest component to take from prototype into production.
- As we saw earlier, it might involve switching from a basic web framework to something more involved (particularly if you need to add user accounts and the like), but you will be able to find an equivalent for whichever language you have chosen.
- That means most of the business logic will move across with minimal changes.

OPEN SOURCE VERSUS CLOSED SOURCE

FAQ

1. Discuss Open source and Closed Source Hardware and Software. State their advantages and disadvantages.

WHY OPEN?

FAQ

1. What are the disadvantages of Open Source?
2. “Open Source has a competitive advantage. “Discuss.

- Open source software refers to the computer software in which source is open i.e. The general public canaccess and use it.
- In short it is referred to as OSS.
- The source code of open source software is public.
- It uses the code freely available on the Internet.
- This code can be modified by other users and organizations i.e. the source code is available for anyone to lookat.
- The open source software is either free or the price is very less.
- There are not so many restrictions on users based on usability and modification of software.
- Some examples of open source software are Firefox, Open Office, Gimp, Alfresco, Android, Zimbra, Thunderbird, MySQL, Mailman, Moodle, TeX, Samba, Perl, PHP, KDE etc.

Advantages of Open Source Software:

- **Tested Software:** Using open source work is often a no-risk way of getting software that has been tested, improved, and debugged by many eyes. As long as it isn’t licensed with an extreme viral license (such as theAGPL), there is no reason not to use such work, even in a closed source project.
- **Mindshare:** Using open source aggressively gives the product the chance to gain mindshare. The design is open;therefore, many other companies have produced clones of the board or components such as shields that are compatible with it.

(Mindshare means consumer awareness of a product or brand, typically as opposed to market share.)

- **Cost:** Open source software is generally free, which means organizations can use it without any licensing fees.
- **Customization:** Since the source code is available, developers can modify and customize the software to meetspecific requirements.
- **Community Support:** Open source software has a large community of users who contribute to documentation,bug fixes, and improvements.
- **Security:** With open source software, security vulnerabilities can be detected and fixed quickly by the community.

- **Transparency:** Since the source code is open, users can see how the software works and what data it collects.
- Disadvantages of Open Source Software:
- **Limited Technical Support:** While there is a large community of users who can help troubleshoot issues, thereis no guarantee of professional technical support.
- **Complexity:** Open source software can be more difficult to set up and configure than closed source software,especially for users who are not experienced in software development.
- **Lack of Features:** Open source software may not have all the features that are available in closed source software, especially for niche or specialized industries.
- **Vulnerability:** Some OSS products can have its coding altered so that those who wish to exploit others can doso. This may include identity theft, virus transfers, and other activities that irritate open source software products.

WHY CLOSED?

- Closed source software refers to the computer software in which source code is closed i.e. the public is notgiven access to the source code.
- In short it is referred to as CSS. In closed source software the source code is protected.
- Only an individual or organization who/that has created the software can change it.
- The price of closed source software is high and users need to have a valid and authenticated license to use thesoftware.
- As it issues an authenticated license it also puts a lot of restrictions on users based on usability and modificationof software.
- Some examples of closed source software are Skype, Google earth, Java, Adobe Flash, Virtual Box, Adobe Reader, Microsoft office, Microsoft Windows, WinRAR, mac OS, Adobe Flash Player etc.

Advantages of Closed Source Software

- **Technical Support:** Closed source software usually comes with professional technical support, which can be helpful for organizations that need assistance with setup, configuration, or troubleshooting.
- **Features:** Closed source software typically has more features than open source software, including advancedanalytics, reporting, and data visualization tools.
- **Security:** Closed source software often has built-in security features and can provide better protection againstcyber threats.
- **Integration:** Closed source software is often designed to work seamlessly with other enterprise software, making integration with existing systems easier.

Disadvantages of Closed Source Software:

- **Cost:** Closed source software can be expensive, with licensing fees and maintenance costs that can add up overtime.
- **Vendor Lock-In:** Organizations that use closed source software may become dependent on the vendor and findit difficult to switch to another software.
- **Limited Customization:** Closed source software may not be as customizable as open source software, which canbe a disadvantage for organizations with specific requirements.
- **Lack of Transparency:** Since the source code is not available, users cannot see how the software works or whatdata it collects, which may raise privacy concerns.

Difference between open source and closed source
FAQ

1. Differentiate between open source and closed source.

Sr.No	Parameters	Open Source Software	Closed Source Software
1	Definition	Open source software refers to the computer software in which source is open i.e. the general public can access and use.	Closed source software refers to the computersoftware in which source code is closed i.e. thepublic is not given access to the source code.
2	Short form	Open Source Software in short is also referred to as OSS.	Closed Source Software in short is also referred to as CSS.
3	Source code Access	The source code of open source software is public.	In closed source software the source code is protected.
4	Editing/Modification Access	This code can be modified by other users and organizations means that the source code is available for anyone to look at.	The only individual or organization who has created the software can only modify the code.
5	Cost	The price of open source software is very low.	The price of closed source software is high.
6	Usability Restrictions	There are not so many restrictions on users based on usability and modification of software.	There are so many restrictions on users based on usability and modification of software.
7	Competition	Programmers compete with each other for recognition.	Programmers do not compete with each other for recognition.
8	Software Improvement	Programmers freely provide improvement for recognition if their improvement is accepted.	Programmers are hired by the software firm/organization to improve the software.
9	No of people working on a project	If the program is popular then a very large number of programmers may work on the project.	There is a limitation on the number of programmers/teams who will work on the project.
10	Purchase package	It is purchased with its source code.	It is not purchased with its source code.
11	Installation	Open software can be installed into any computer.	Closed software needs to have a valid license before installation into any computer.
12	Failure	Open source software fails fast and fixes faster.	Closed source software has no room for failure.

Similarities between Open Source Software and Closed Source Software:

- Both can be used to perform a wide range of tasks and support a variety of applications.
- Both can be designed to work on multiple operating systems, including Windows, Linux, and macOS.
- Both can be used to support mission-critical applications and services.
- Both can be optimized for performance, scalability, and security.
- Both can be accessed and managed remotely using a variety of tools and interfaces.
- Both can be updated and maintained regularly to fix bugs, add new features, and improve performance.

Mixing open and closed source
FAQ

1. Discuss the merits and demerits of mixing open and closed source softwares.

- Mixing of Open Source and Closed Source allows us to get the advantages of both Open Source and Closed Source.
- Mixing open source and closed source enables to open source your libraries and keeping your core businessclosed
- Example of Mixing Open and Closed Source is Adrian's project Bubblino has a mix of licenses:
- Arduino code is open source.
- Schematics are available but not especially well advertised.
- Server code is closed source.

CLOSED SOURCES FOR MASS MARKET PROJECTS

- When you can realistically expect that a project might be not just successful but huge, that is, a mass marketcommodity.
- If you could get an existing supply and distribution chain on your side, the advantage of being first to market and doing so cheaper may well be the most important thing.
- Let's consider Nest, an intelligent thermostat: the area of smart energy metering and control is one in whichmany people are experimenting.
- The moment that an international power company chooses to roll out power monitors to all its customers, sucha project would become instantaneously mass market.
- This would make it a very tempting proposition to copy, if you are a highly skilled, highly geared-up manufacturer in China, for example.
- If you also have the schematics and full source code, you can even skip the investment required to reverse-engineer the product.
- The costs and effort required in moving to mass scale show how, for a physical device, the importance of supplychain can affect other considerations.

TAPPING INTO THE COMMUNITY
FAQ

1. How can one tap into the community for promoting IOT devices? Explain.

- While thinking about which platform you want to build for, having a community to tap into may be vital or atleast useful.
- If you have a problem with a component or a library, or a question about how to do something you could simply do a Google search on the words "Arduino servo potentiometer" and find a YouTube video, a blog post, or some code.
- If you are doing something more obscure or need more detailed technical assistance, finding someone who hasalready done exactly that thing may be difficult.
- When you are an inexperienced maker, using a platform in which other people can mentor you is invaluable.
- Local meetings are also a great way to discuss your own project and learn about others.
- While to discuss your project is in some way being "open" about it, you are at all times in control of how muchyou say and whom you say it to.
- In general, face-to-face meetings at a hack space may well be a friendlier and more supportive way to dip yourtoes into the idea of a "community" of Internet of Things makers.

PROTOTYPING EMBEDDED DEVICES

ELECTRONICS

- Most of the prototyping can be done on what are called ***solderless breadboards.***
- They enable you to build components together into a circuit with just a push-fit connection.
- When it comes to thinking about the electronics, it's useful to split them into two main categories:
- **Sensors:** Sensors are the ways of getting information *into* your device, finding out things about your surroundings.
- **Actuators:** Actuators are the *outputs* for the device—the motors, lights, and so on, which let your device dosomething to the outside world.
- Within both categories, the electronic components can talk to the computer in a number of ways.
- The simplest is through digital I/O, which has only two states: a button can either be pressed or not; or an LEDcan be on or off.
- These states are usually connected via general-purpose input/output (GPIO) pins and map a digital 0 in the processor to 0 volts in the circuit and the digital 1 to a set voltage.
- Usually the voltage that the processor is using to run (commonly 5V or 3.3V)
- If you want a more nuanced connection than just on/off, you need an analogue signal.
- If you want to run a motor at a speed other than off or full-speed, you need to feed it with a voltage somewherebetween 0V and its maximum rating.
- Because computers are purely digital devices, you need a way to translate between the analogue voltages in thereal world and the digital of the computer.
- An analogue-to-digital converter (ADC) lets you measure varying voltages.
- Microcontrollers often have a number of these converters built in.
- They will convert the voltage level between 0V and a predefined maximum into a number, depending on theaccuracy of the ADC.
- The flipside of an ADC is a DAC, or digital-to-analogue converter.
- DACs let you generate varying voltages from a digital value.

SENSORS AND ACTUATORS

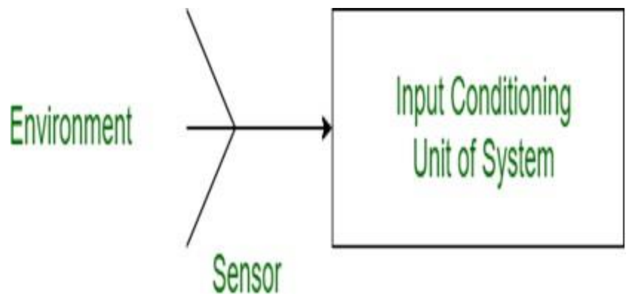
FAQ

1. Write a note on sensors and actuators.

- Actuators and sensors are two essential components of embedded and electronic systems.
- Actuators form a link with the output ports while sensors connect to the input ports of a given system.
- These are used in several real-life applications such as flight control systems in an aircraft, process control systems in nuclear reactors, power plants that require to be operated on an automated control.
- Both actuators and sensors play a significant role in condition-based maintenance.
- These devices act as the mediator between the physical environment and the electronic system where the sensor and actuator are embedded.
- Although they frequently work together, actuators and sensors differ from each other in function and purposes

Sensor

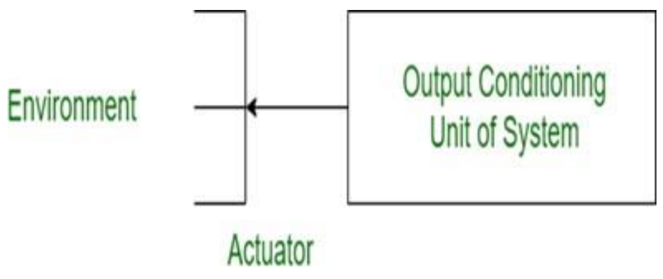
- Sensor is an object whose purpose is to detect events or changes in its environment, and then provide a corresponding output.
- A sensor is a type of transducer, sensors may provide various types of output, but typically use electrical or optical signals.
- Sensors are placed at the input port and act as inputs
- There is a range of sources, including light, temperature, and pressure etc can be detected.
- For example, a thermometer takes the temperature as physical characteristic and then converts it into electricalsignals for the system.



- Some commonly used sensors are:
    - Temperature sensor
    - Humidity sensor
    - Pressure sensor
    - Ultrasonic sensor
    - Proximity sensor
    - Gas sensor
    - Smoke detector

Actuator

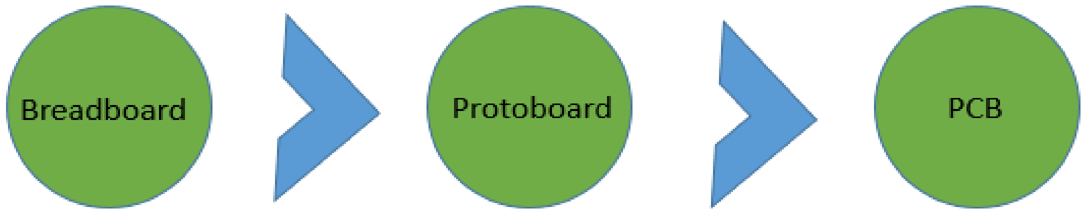
- An actuator is a device that takes an electrical signal and blends it with an energy source to create movement.
- It is a component of machines that is responsible for moving or controlling a mechanism or system.
- Actuator is used for output.
- It is a transducer that converts either mechanical or electrical signal into physical signals.
- For example, An LED is a device that converts electrical energy into light energy.



SCALING UP THE ELECTRONICS
FAQ
1. With the help of an example, explain the process of scaling up the electronics.

- From the perspective of the electronics, the starting point for prototyping is usually a "breadboard".
- This lets you push-fit components and wires to make up circuits without requiring any soldering and therefore makes experimentation easy.
- When you're happy with how things are wired up, it's common to solder the components onto some protoboard, which may be sufficient to make the circuit more permanent and prevent wires from going astray.
- Moving beyond the protoboard option tends to involve learning how to lay out a PCB.
- For small production runs, you'll likely use through-hole components, so called because the legs of the component go through holes in the PCB and tend to be soldered by hand.
- You will often create your designs as companion boards to an existing microcontroller platform—generally called shields.
- This approach lets you bootstrap production without worrying about designing the entire system from scratch.
- When you want to scale things even further, moving to a combined board allows you to remove any unnecessary components from the microcontroller board, and switching to surface mount components—wherethe legs of the chips are soldered onto the same surface as the chip—eases the board's assembly with automated manufacturing lines.

Example
- Let's look at the evolution of part of the Bubblino circuitry, from initial testing, through prototype, to finishedPCB:
- The first step in creating your circuit is generally to build it up on a breadboard.
- This way, you can easily reconfigure things as you decide exactly how it should be laid out on the breadboard.
- When you are happy with how the circuit works, soldering it onto a stripboard will make the layout permanent.
- This means you can stop worrying about one of the wires coming loose, and if you're going to make only onecopy of the circuit, that might be as far as you need to take things.
- If you need to make many copies of the circuit, or if you want a professional finish, you can turn your circuit intoa PCB.
- This makes it easier to build up the circuit because the position of each component will be labeled, there will beholes only where the components go, and there will be less chance of short circuits because the tracks between components will be protected by the solder resist.



EMBEDDED COMPUTING BASICS
Microcontrollers

- These systems combine the processor, RAM, and storage onto a single chip, which means they are much more
- specialized, smaller than their PC equivalents, and also easier to build into a custom design.
- Microcontrollers are very limited in their capabilities.
- Usually, they offer RAM capabilities measured in kilobytes and storage in the tens of kilobytes.
- The modern chips are much smaller, require less power, and run about five times faster than their 1980s counterparts.
- The microcontroller market consists of many manufacturers each with a range of chips for different applications.
- (Atmel, Microchip, NXP, Texas Instruments, to name a few).
- The devices using them are focused on performing one task.

System-on-chips:

- In between the low-end microcontroller and a full-blown PC sits the SoC.
- Like the microcontroller, these SoCs combine a processor and a number of peripherals onto a single chip butusually have more capabilities.
- The processors usually range from a few hundred megahertz to gigahertz.
- RAM is measured in megabytes rather than kilobytes.
- Storage for SoC. modules tends not to be included on the chip, with SD cards being a popular solution.
- The greater capabilities of SoC. mean that they need some sort of operating system to marshal their resources.
- A wide selection of embedded operating systems, both closed and open source, is available from both specialized embedded providers and the big OS players, such as Microsoft and Linux.

CHOOSING YOUR PLATFORM
FAQ

1. Explain the following with respect to prototyping embedded devices: Processor Speed, RAM, Networking, Power Consumption and Physical Size & Form Factor.
2. Discuss the factors we should consider when deciding to build an IOT Device.

- The platform you choose depends on the particular blend of price, performance, and capabilities that suit whatyou’re trying to achieve.
- Start by choosing a platform to prototype in.
- Some of the factors that you need to weigh when deciding how to build your device:

Processor Speed

- The processor speed, or clock speed, of your processor tells you how fast it can process the individual instructions in the machine code for the program it’s running.
- Naturally, a faster processor speed means that it can execute instructions more quickly.
- Some processors may lack hardware support for floating-point calculations, so if the code involves a lot of complicated mathematics, a slower processor with hardware floating-point support could be faster than a slightly higher performance processor without it.
- If your project doesn’t require heavyweight processing then some sort of microcontroller will be fast enough.
- Microcontrollers tend to be clocked at speeds in the tens of MHz, whereas SoCs run at hundreds of MHz or possibly low GHz.

RAM

- RAM provides the working memory for the system.
- If you have more RAM, you may be able to do more things or have more flexibility over your choice of codingalgorithm.
- It is difficult to give exact guidelines to the amount of RAM you will need, as it will vary from project to project.
- However, microcontrollers with less than 1KB of RAM are unlikely to be of interest, and if you want to run standard encryption protocols, you will need at least 4KB, and preferably more.
- For SoC. boards, particularly if you plan to run Linux as the operating system, we recommend at least 256MB.

Networking

- How your device connects to the rest of the world is a key consideration for Internet of Things products.
- Wired Ethernet is often the simplest for the user—generally plug and play—and cheapest, but it requires a physical cable.
- Wireless solutions obviously avoid that requirement but introduce a more complicated configuration.
- Wi-Fi is the most widely deployed to provide an existing infrastructure for connections, but it can be more expensive and less optimized for power consumption than some of its competitors.
- Other short-range wireless can offer better power-consumption profiles or costs than Wi-Fi but usually with thetrade-off of lower bandwidth
- There is, of course, the existing Bluetooth standard as another possible choice.
- For remote or outdoor deployment the mobile phone networks can be used.

USB

- If your device can rely on a more powerful computer being nearby, tethering to it via USB can be an easy way toprovide both power and networking.
- You can buy some of the microcontrollers in versions which include support for USB.
- Instead of the microcontroller presenting itself as a device, some can also act as the USB "host".

Power Consumption

- Faster processors are often more power hungry than slower ones.
- For devices which might be portable or rely on an unconventional power supply (batteries, solar power) depending on where they are installed, power consumption may be an issue.
- However, processors may have a minimal power-consumption sleep mode.
- This mode may allow you to use a faster processor to quickly perform operations and then return to low-powersleep.

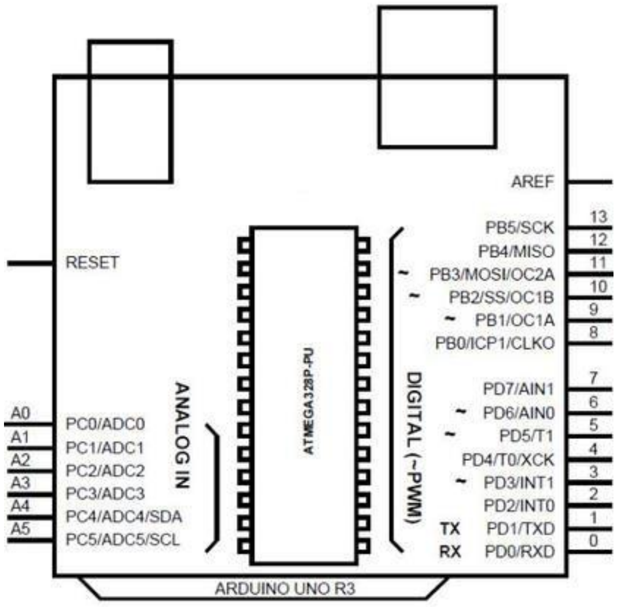
Interfacing with Sensors and Other Circuitry

- In addition to talking to the Internet, your device needs to interact with something else—either sensors to gather data about its environment; Or motors, LEDs, screens, and so on, to provide output.
- You could connect to the circuitry through some sort of peripheral bus—SPI and I2C being common ones—orthrough ADC or DAC modules to read or write varying voltages; or through generic GPIO pins, which provide digital on/off inputs or outputs.

Physical Size and Form Factor

- The continual improvement in manufacturing techniques for silicon chips means that we've long passed the point where the limiting factor in the size of a chip is the amount of space required for all the transistors andother components that make up the circuitry on the silicon.
- Nowadays, the size is governed by the number of connections it needs to make to the surrounding componentson the PCB.
- The limit to the size that each connection can be reduced to is then governed by the capabilities and tolerancesof your manufacturing process.
- Some surface-mount designs are big enough for home-etched PCBs and can be hand soldered.
- Others require professionally produced PCBs and accurate pick-and-place machines to locate them correctly.
- Due to these trade-offs in size versus manufacturing complexity, many chip designs are available in a number ofdifferent form factors, known as packages.
- This lets the circuit designer choose the form that best suits his particular application.

ARDUINO



Introduction

- Arduino board is an open-source platform used to make electronics projects.
- It consists of both a microcontroller and a part of the software or Integrated Development Environment (IDE)that runs on your PC, used to write & upload computer code to the physical board.
- There are many types of Arduino. Some of them are:
  - Arduino Uno (R3)
  - Arduino Nano
  - Arduino Micro
  - Arduino Due
  - LilyPad Arduino Board
  - Arduino Bluetooth
  - Arduino Mega (R3) Board
  - Arduino Leonardo Board
  - Arduino Ethernet

- The Arduino team’s focus on simplicity rather than raw performance for the code has made the Arduino theboard of choice in almost every beginner’s physical computing project.
- The most commonly used Arduino is ARDUINO UNO(R3).
- Some features of Arduino Uno are as follows:
  - 14 Digital In/Out pins (6 can be used as PWM)
  - 6 Analog Inputs
  - A USB Connection
  - A Power Jack
  - Reset Button
  - On-board LED
  - SCL/SDA pins
- Arduino Mega 2560 provides 256KB of Flash storage, 8KB of RAM, three more serial ports, and a massive 54 GPIO pins.
- The more recent Arduino Due has a 32-bit ARM core microcontroller.
- Its specs are similar to the Mega’s, although it ups the RAM to 96KB.

DEVELOPING ON THE ARDUINO
FAQ

1. How is development done for Arduino? Explain
2. Describe Arduino with a focus on the following aspects: Integrated Development Environment, Pushing Code, Operating System, Programming Language and Openness.

- Using a single USB cable, you can not only power the board but also push your code onto it, and (if needed)communicate with it.
- We develop a project in Arduino using the following tools/ steps:

Integrated Development Environment

- You usually develop against the Arduino using the integrated development environment (IDE) that the teamsupplies at http://arduino.cc.
- Most Arduino projects consist of a single file of code, so you can think of the IDE mostly as a simple file editor.
- The controls that you use the most are those to check the code (by compiling it) or to push code to the board.

Pushing Code

- Connecting to the board should be relatively straightforward via a USB cable.
- You need to choose the correct serial port and the board type (look carefully at the labeling on your board andits CPU to determine which option to select).
- When your setup is correct, the process of pushing code is generally simple:
- First, the code is checked and compiled, with any compilation errors reported to you.
- If the code compiles successfully, it gets transferred to the Arduino and stored in its flash memory.
- At this point, the Arduino reboots and starts running the new code.

Operating System

- The Arduino doesn't, by default, run an OS as such, only the boot loader, which simplifies the code-pushingprocess described previously.
- When you switch on the board, it simply runs the code that you have compiled until the board is switched offagain (or the code crashes).
- It is, however, possible to upload an OS to the Arduino, usually a lightweight real-time operating system (RTOS)such as FreeRTOS/DuinOS.
- The main advantage of one of these operating systems is their built-in support for multitasking.

Language

- The language usually used for Arduino is a slightly modified version of C++.
- It includes some libraries used to read and write data from the I/O pins provided on the Arduino and to do somebasic handling for "interrupts".
- The code needs to provide only two routines:
    - setup ():
        - This routine is run once when the board first boots.
        - You could use it to set the modes of I/O pins to input or output or to prepare a data structure which will beused throughout the program.
    - loop ():
        - This routine is run repeatedly in a tight loop while the Arduino is switched on.
        - Typically, you might check some input, do some calculation on it, and perhaps do some output in response.

Debugging

- Because C++ is a compiled language, a fair number of errors, such as bad syntax or failure to declare variables,are caught at compilation time.
- Because the Arduino isn't generally connected to a screen, it is hard for it to tell you when something goeswrong.
- Even if the code is compiled successfully, certain errors still happen.
- An error could be raised that can't be handled, such as a division by zero, or trying to access the tenth elementof a 9-element list.
- In the absence of a screen, the Arduino allows you to write Information over the USB cable using Serial. Write ().
- The Arduino IDE provides a serial monitor which echoes the data that the Arduino has sent over the USB cable.
- This could include any textual information, such as logging information, comments, and details about the datathat the Arduino is receiving and processing (to double-check that your calculations are doing the right thing).

Some notes on the hardware
Headers

- The Arduino exposes a number of GPIO pins and is usually supplied with "headers" (plastic strips that sit on thepin holes that provide a convenient solderless connection for wires, especially with a "jumper" connection).
- The headers are optimized for prototyping and for being able to change the purpose of the Arduino easily.

Pins on the Arduino

- Each pin is clearly labeled on the controller board.
- In general, the Arduino has numbered digital pins, and numbered analogue pins prefixed with an A.
- The details of pins vary from the smaller boards such as the Nano, the classic form factor of the Uno, and thelarger boards such as the Mega or the Due.

Power Connections

- In general, you have power outputs such as 5 volts or 3.3 volts (usually labeled 5V and 3V3, or perhaps just 3V),
- There are one or more electric ground connections (GND) in any Arduino.
- You can power the Arduino using a USB connection from your computer.
- This capability is usually quite convenient during prototyping because you need the serial connection in any caseto program the board.
- The Arduino also has a socket for an external power supply, which you might be more likely to use if you distribute the project.
- Either way should be capable of powering the microcontroller and the usual electronics that you might attach toit.
- In the case of larger items, such as motors, you may have to attach external power and make that availableselectively to the component using transistors.
- Outside of the standard boards, a number of them are focused on a particular niche application—for example,the Arduino Ethernet has an on-board Ethernet chip and trades the USB socket for an Ethernet one, making it easier to hook up to the Internet.
- This is obviously a strong contender for a useful board for Internet of Things projects.
- The LilyPad has an entirely different specialism, as it has a flattened form (shaped, as the name suggests, like aflower with the I/O capabilities exposed on its “petals”) and is designed to make it easy to wire up with conductive thread, and so a boon for wearable technology projects.

Shields

- Choosing one of the specialist boards isn’t the only way to extend the capabilities of your Arduino.
- Most of the boards share the same layout of the assorted GPIO, ADC, and power pins, and you are able to piggyback an additional circuit board on top of the Arduino which can contain all manner of componentry to give the Arduino extra capabilities.
- In the Arduino world, these add-on boards are called shields, perhaps because they cover the actual board as ifprotecting it.
- Some shields provide networking capabilities—Ethernet, Wi-Fi, or Zigbee wireless, for example.
- Motor shields make it simple to connect motors and servos;
- there are shields to hook up mobile phone LCD screens;
- others to provide capacitive sensing; others to play MP3 files or WAV files from an SD card; and all manner ofother possibilities—so much so that an entire website, http://shieldlist.org/, is dedicated to comparing and documenting them.
- In terms of functionality, a standard Arduino with an Ethernet shield is equivalent to an Arduino Ethernet. However, the latter is thinner (because it has all the components laid out on a single board) but loses the convenient USB connection. (You can still connect to it to push code or communicate over the serial connectionby using a supplied adapter.)

Openness

- The Arduino project is completely open hardware and an open hardware success story.
- The only part of the project protected is the Arduino trademark, so they can control the quality of any boardscalling themselves an Arduino.
- In addition to the code being available to download freely, the circuit board schematics and even the EAGLEPCB design files are easily found on the Arduino website.
- This culture of sharing has borne fruit in many derivative boards being produced by all manner of people.
- Some are merely minor variations on the main Arduino Uno, but many others introduce new features or formfactors that the core Arduino team have overlooked.

FAQ

1. Explain the following IOT Devices built with Arduino.
   i. The Good Night Lamp ii. Botanicals iii. Baker Treat

The Good Night Lamp

- The idea of Good Night Lamp was initially brought up by Alexandro Deschamps Soncino.
- He came up with the idea of an internet connected table or bedside lamp.
- A simple consumer device, this lamp would be paired with another lamp anywhere in the world, allowing it toswitch the other lamp on or off, and vice versa.
- Because light is integrated into our daily routine, seeing when our loved ones turn, for example, their bedside lamp on or off gives us a calm and ambient view onto their lives.
- The product consists of a “big lamp” which is paired with one or more “little lamps”.
- The big lamp has its own switch and is designed to be used like a normal lamp.
- The little lamps, however, don’t have switches but instead reflect the state of the big lamp.
- Adrian and the rest of the team’s familiarity with Arduino led to it being an obvious choice as the prototypingplatform.
- In addition, as the lamps are designed to be a consumer product rather than a technical product, and are targeted at a mass market, design, cost, and ease of use are also important.
- The Arduino platform is simple enough that it is possible to reduce costs and size substantially by choosing which components you need in the production version.
- A key challenge in creating a mass-market connected device is finding a convenient way for consumers, some ofwhom are non-technical, to connect the device to the Internet.
- Even if the user has Wi-Fi installed, entering authentication details for your home network on a device that hasno keyboard or screen presents challenges.
- As well as looking into options for the best solution for this issue, the Good Night Lamp teams are also building aversion which connects over the mobile phone networks via GSM or 3G.
- This option fits in with the team’s vision of connecting people via a “physical social network”, even if they arenot otherwise connected to the Internet.

Botanicals

- Botanicals (www.botanicalls.com/) is a collaboration between technologists and designers that consists of monitoring kits to place in plant pots.
- The Botanicals kits then contact the owner if the plant’s soil gets too dry.
- The project write-up humorously refers to this as “an effort to promote successful interspecies understanding”and as a way of translating between a plant’s communications protocols (the color and drooping of leaves) to human protocols, such as telephone, email, or Twitter.
- The original project used stock Arduino controllers, although the kits available for sale today use the AT mega 168 microcontroller with a custom board, which remains Arduino-compatible, and the programming

is all done using the Arduino IDE.
- To match the form factor of the leaf-shaped printed circuit board (PCB), the device uses a Wiz Net Ethernet chipinstead of the larger Arduino Ethernet Shield.
- Future updates might well support Wi-Fi instead.

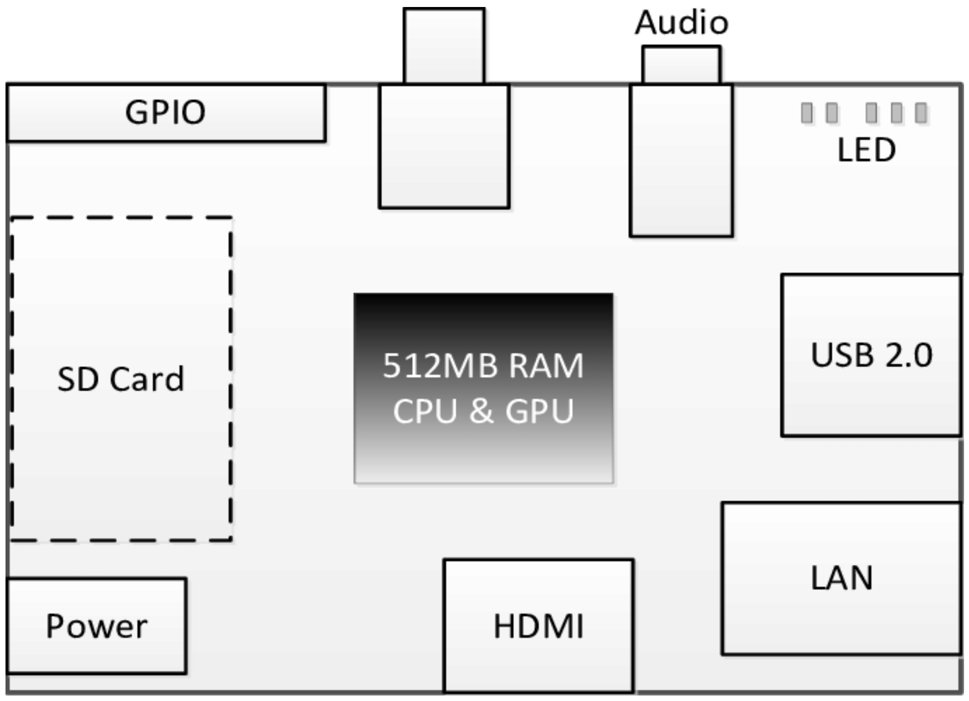
Baker Tweet

- The Baker Tweet device (www.bakertweet.com/) is effectively a physical client for Twitter designed for use in a bakery.
- A baker may want to let customers know that a certain product has just come out of the ovens—fresh bread, hot muffins, cupcakes laden with icing—yet the environment he would want to tweet from contains hot ovens, flour dust, and sticky dough and batter, all of which would play havoc with the electronics, keyboard, and screenof a computer, tablet, or phone.
- Staff of design agency Poke in London wanted to know when their local bakery had just produced a fresh batchof their favorite bread and cake, so they designed a proof of concept to make it possible. Because Baker Tweet communicates using Wi-Fi, bakeries, typically not built to accommodate Ethernet cables, can install it.
- Baker Tweet exposes the functionality of Twitter in a “bakery-proof” box with more robust electronics than ageneral-purpose computer, and a simplified interface that can be used by fingers covered in flour and dough.
- It was designed with an Arduino, an Ethernet Shield, and a Wi-Fi adapter.
- As well as the Arduino simply controlling a third-party service (Twitter), it is also hooked up to a custom servicewhich allows the baker to configure the messages to be sent.

Raspberry Pi
FAQ

1. Write a note on Raspberry Pi.

- The Raspberry Pi is effectively a computer that can run a real, modern operating system, communicate with a keyboard and mouse, talk to the Internet, and drive a TV/monitor with high-resolution graphics.
- Raspberry Pi is a credit card-sized, low-cost but fully functional and programmable computer with modern high definition multimedia capabilities
- It may be the device that gets us back to computing basics.
- The Raspberry Pi device looks like a motherboard, with the mounted chips and ports exposed (something you'd expect to see only if you opened up your computer and looked at its internal boards), but it has all the components you need to connect input, output, and storage devices and start computing.
- Here are the various components on the Raspberry Pi board:
    - ARM CPU/GPU: The CPU performs all basic computations and the GPU performs graphics related functions.
    - GPIO: These are exposed general-purpose input/output connection points that will allow the real hardwarehobbyists the opportunity to tinker.
    - RCA: Used for connecting analog devices.
    - Audio out: It is a 3.5mm audio output jack.
    - LEDs: Used for giving some notification.
    - USB: Used for connecting some peripherals or even cascading the USB ports.
    - HDMI: Used for connecting HD devices.
    - Power: Used for power supply.
    - SD card slot: It has an OS in it used while booting.
    - Ethernet: Used for wired LAN.



Cases and extension boards

- Several ecosystems have built up around the device.
- Because the Pi can be useful as a general-purpose computer or media center without requiring constant prototyping with electronic components, one of the first demands enthusiasts have had was for convenient and attractive cases for it.
- Many makers blogged about their own attempts and have contributed designs to Thingiverse, Instructables, and others.
- There have also been several commercial projects.
- Beyond these largely aesthetic projects, extension boards and other accessories are already available for the Raspberry Pi.
- Many interesting kits are in development, such as the Gertboard designed for conveniently playing with the GPIO pins. Gertboard is an input/output (I/O) extension board for the Raspberry Pi computer. It comes with a large variety of components, including buttons, LEDs, A/D and D/A converters, a motor controller, and an Atmel AVR microcontroller
- A lot of people are doing interesting things with their Pis, but as the platform is so much more high level andcapable, the attention may be spread more thinly—from designing cases to porting operating systems to working on media center plug-ins.

DEVELOPING ON THE RASPBERRY PI

- We develop a project in Arduino using the following tools/ steps:
- Operating System
- Although many operating systems can run on the Pi, we recommend using a popular Linux distribution, such as
  - **Raspbian:** Released by the Raspbian Pi Foundation, Raspbian is a distro based on Debian. This is the default “official” distribution and is certainly a good choice for general work with a Pi. Raspbian requiresa brief configuration stage first.
  - **Occidentalis:** This is Adafruit’s customized Raspbian. Unlike Raspbian, the distribution assumes that youwill use it “headless”—not connected to keyboard and monitor—so you can connect to it remotely by default.

Programming Language
FAQ

1. Compare python characteristics with C++.

- One choice to be made is which programming language and environment you want to use.
- Here, again, there is some guidance from the Foundation, which suggests Python as a good language for educational programming (and indeed the name “Pi” comes initially from Python).
- Some advantages of using Python are
    - Easy to learn, read and maintain
        - Python is a minimalistic language with relatively few keywords, uses EnglishKeywords and has fewer syntactical constructions as compared to other languages.
        - Reading Python programs feels like English with pseudo-code-like constructs.
        - Python is easy to learn yet an extremely powerful language for a wide range ofapplications.
    - Object and Procedure Oriented
        - Python supports both procedure oriented programming and object oriented programming.
        - Procedure oriented paradigm allows programs to be written around objectsthat include both data and functionality.
        - Procedure oriented paradigm allows programs to be written around proceduresor functions that allow reuse of code.
    - Extendable
        - Python is an extendable language and allows integration of low level modules written in languages such as C/C++.
        - This is useful when you want to speed up a program
    - Scaleable
        - Due to the minimalistic nature of Python, it provides manageable structure for large programs.
    - Portable
        - Since python is an interpreted Language, programmers do not have to worry about compilation, linking and loading of programs.
        - Python programs can be directly executed from the source.
    - Broad Library Support
        - Python has a broad library support and works on various platforms such asWindows, Linux, Mac etc

Comparison of python characteristics with C++

Sr.No	Parameters	Python	C++
1	Code	Python has fewer lines of code	C++ tends to have long lines of code
2	Compilation	Python is interpreted	C++ is precompiled
3	Speed	It is slower since it uses an interpreter and also determines the data type at run time	It is faster once compiled as compared to python
4	Efficiency	Specialized formatting not common in other languages, script-like language, OOP features, code reuse through libraries	C-like syntax, powerful OOP features and operator overloading, best compile-time optimizer
5	Nature	It is dynamically typed.	It is statically typed
6	Functions	Python Functions do not have restrictions on the type of the argument and the type of its return value	In C++, the function can accept and return the type of value which is already defined

7	Scope of Variable	Variables are accessible even outside the loop	The scope of variables is limited within the loops
8	Extension	Python programs are saved with the .py extension	C++ programs are saved with the .cpp extension
9	Popularity	It has huge community support. When it comes to popularity, beginner and novice programmers tend to turn to Python	It also has dedicated followings online. But only the people who have some experience in the field show a lot of interest in C++
10	Garbage Collection	It supports garbage collection	It doesn't support garbage collection, but it can be implemented
11	Rapid Prototyping	Rapid Prototyping is possible, with easy project setup, live interpreter	Rapid Prototyping is possible, but project setup can be complicated, live interpreter through IRC bot
12	Variable Declaration	Variables are declared by simply writing their name with their data type	While declaring a variable it is necessary to mention its data type

Debugging

- While Python's compiler also catches a number of syntax errors and attempts to use undeclared variables, it isalso a relatively permissive language (compared to C++) which performs a greater number of calculations at runtime.
- This means that additional classes of programming errors won't cause failure at compilation but will crash theprogram when it's running, perhaps days or months later.
- Python code on Linux gives you the advantages of both the language and the OS.
- One could step through the code using Python's integrated debugger, attach to the process using the Linux's trace command, view logs, and see how much memory is being used, and so on.
- As long as the device itself hasn't crashed, you may be able to ssh into the Raspberry Pi and do some of thisdebugging while your program has failed (or is running but doing the wrong thing).
- For example, one would typically take the opportunity to log details of the error (to help the debugging process)and see if the unexpected problem can be dealt with so that one can continue running the code.
- Python and other high-level languages also have mature testing tools which allow you to assert expected behaviors of your routines and test that they perform correctly.
- This kind of automated testing is useful when you're working out whether you've finished writing correct code, and also can be rerun after making other changes, to make sure that a fix in one part of the code hasn't caused aproblem in another part that was working before.

Some notes on the hardware

- The Raspberry Pi has 8 GPIO pins, which are exposed along with power and other interfaces in a 2-by-13 block ofmale header pins.
- The block of pins provides both 5V and 3.3V outputs.
- However, the GPIO pins themselves are only 3.3V tolerant.
- The Pi doesn't have any over-voltage protection, so you are at risk of breaking the board if you supply a 5V input
- The alternatives are to either proceed with caution or to use an external breakout board that has this kind of protection.
- The Raspberry Pi doesn't have any analogue inputs (ADC), which means that options to connect it to electronicsensors are limited, out of the box, to digital inputs (that is, on/off inputs such as buttons).
- To get readings from light-sensitive photocells, temperature sensors, potentiometers, and so on, you need to connect it to an external ADC via the SPI bus.

- Although it is powered by a standard USB cable, the voltage transmitted over USB from a laptop computer, apowered USB hub, or a USB charger varies greatly.
- If you're not able to power or to boot your Pi, check the power requirements and try another power source.

Openness

- Many of the components are indeed highly open: the customized Linux distributions such as "Raspbian" (basedon Debian), the ARM Video Core drivers, and so on.
- The core Broadcom chip itself is a proprietary piece of hardware, and they have released only a partial datasheet for the BCM2835 chipset.
- However, many of the Raspberry Pi core team are Broadcom employees and have been active in creating driversand the like for it, which are themselves open source.
- These team members have been able to publish certain materials, such as PDFs of the Raspberry Pi board schematics, and so on. However, the answer to the question "Is it open hardware?" is currently "Not yet"

BEAGLEBONE BLACK
FAQ

1. Describe Beagle bone Black with a focus on the following aspects: Operating System, Programming Language, Debugging, Hardware and Openness

- The Beagle Bone Black Board is a low-power open-source single-board computer produced by Texas Instruments.
- These boards are very much designed with the expectation that they will be used for physical computing and experimentation in electronics.
- The original Beagle Bone has no video or audio outputs built in, but it does have a far larger number of GPIO pins, extracted into two rows of female headers.
- It also has the crucial ADC pins for analogue input.
- The Beagle Bone is a computer that mostly runs Linux but is capable of running a variety of other ported operating systems.

Cases and Extensions

- A fair number of attractive cases are available, either sold commercially like the Adafruit BoneBox, or as freely available instructions and designs.
- In any case, given that the board is mostly designed to be used for open hardware, it is also likely that the specific project you are working on will require a custom casing to fit the form factor of the project.
- Extension boards for the BeagleBone are known as "capes" rather than "shields" (the term for Arduino extension boards).
- Capes are available to add controllers for LCD displays and motors and various networking and sensor applications.

DEVELOPING ON THE BEAGLEBONE
Operating System

- The BeagleBone Black comes pre-installed with Ångström Linux, which is optimized for embedded systems. This distribution is lightweight and designed to run "headless," meaning without a monitor or keyboard, which suits its role in embedded applications.
- Ångström supports a variety of development environments and can be tailored to specific project needs.

Programming Language

- Programming on the BeagleBone Black primarily revolves around JavaScript, particularly with Node.js. Node.js is chosen for its event-driven, non-blocking I/O model, which is advantageous for handling data-intensive real-time applications.
- The Cloud9 IDE, hosted either locally or online, provides a convenient environment for developing Node.js applications directly on the board.

Debugging

- Debugging on the BeagleBone Black involves leveraging the tools available in the Linux operating system.
- Developers have access to standard Linux debugging utilities, which are essential for diagnosing and fixing issues in complex applications.
- The Cloud9 IDE also supports runtime error reporting, aiding in the identification of errors during application execution.

Some Notes on the Hardware

- **GPIO and I/O**: The BeagleBone Black excels in hardware interfacing capabilities with its 65 GPIO pins, including support for PWM and ADC. These features make it highly suitable for physical computing and interfacing with a wide range of sensors and actuators.
- **Expansion Capabilities**: Hardware expansion on the BeagleBone Black is facilitated through "capes," similar to shields on Arduino boards. Capes enable additional functionalities like LCD displays, motor controllers, and various sensors, expanding the board's versatility in IoT and embedded projects.

Openness

- **Open Hardware**: BeagleBone Black is committed to open-source principles, with all hardware designs released under open licenses. This openness allows developers to freely access schematics, bill-of-materials, and PCB layouts, facilitating customization and modification for specific project needs.
- **Open Software**: The Ångström Linux distribution and associated software are also open-source, encouraging a collaborative development environment. The platform supports community-driven improvements and adaptations, ensuring ongoing support and innovation.

Comparison between Arduino and Raspberry pi

FAQ

1. Compare Raspberry Pi and Arduino

Sr.No	Parameters	Arduino	Raspberry Pi
1	Control Unit	Microcontroller based development board	Microprocessor based development board
2	IC Family	Atmega Series	ARM Processor: Broadcom Chips
3	Use	The Arduino basically helps in controlling all the electrical components that connectto a system's circuit board.	The Raspberry Pi primarily computes data and info for producing valuable outputs. It also controls the various components in any given system on the basis of the outcome (of the computation).
4	Openness	Arduino is an open-source electronics platform that uses simple hardware and software to make it easy.	Raspberry Pi's hardware and software are bothproprietary. The Raspberry Pi has never claimed to be open source. Many aspects of it,particularly the software, are open source, but not all of it is.

5	Structure of Hardware and Software	The Arduino boards have a very simple structure of software and hardware.	The Raspberry Pi boards consist of comparatively complex software and hardware architecture.
6	Type of CPU Architecture	Arduino has an 8-bit architecture.	Raspberry Pi has a 64-bit architecture.
7	Operating System	Absent	Present
8	Cost Efficiency	It has a higher cost-efficiency because it is comparatively cheaper.	It has a lower cost-efficiency because it is comparatively more expensive.
9	Logic Level	It works at a Logic Level of 5V	It works at a Logic Level of 3.3V
10	ADC	Inbuilt ADC present	No Inbuilt ADCs
11	I/O Drive Strength	The I/O current drive strength in the caseof Arduino is higher.	The I/O current drive strength in the case of Raspberry Pi is lower.
12	RAM Usage	Arduino makes use of very little RAM of about 2 kB (Kilobytes).	Raspberry Pi always requires more RAM than Arduino of about 1 GB (Gigabytes).
13	Processing Speed	Arduino clocks 16 MHz (Megahertz) of processing speed in a system.	The Raspberry Pi clocks 1.4 GHz (Gigahertz) processing speed in a system.
14	Power Consumption	Arduino consumes power of about 200 MW (Megawatts).	Raspberry Pi consumes about 700 MW.

Comparison between BeagleBone Black and Raspberry pi
FAQ
1. Compare Raspberry pi and Beagle bone black.

Sr.No	Parameters	BeagleBone Black	Raspberry Pi Model B
1	CPU Speed	1GHz ARM Cortex-A8	700 MHz ARM11
2	GPU	SGX530 3D	Broadcom Dual-Core Video Core IV Media Co-Processor
3	RAM	512MB	512MB
4	Storage	2GB embedded MMC, plus uSD card	SD card (4GB +)
5	OS	Various Linux distributions, Android, other operating systems available	Various Linux distributions, other operating systems available
6	Connection	65 GPIO pins, of which 8 have PWM 4 UARTs SPI bus I^2C bus USB client + host Ethernet Micro-HDMI out 7 analog inputs (ADC) CAN bus	8 GPIO pins, of which 1 has PWM 1 UART SPI bus with two chip selects I^2C bus 2 USB host sockets Ethernet HDMI out Component video and audio out

Electric Imp
FAQ
1. Write a short note on Electric Imp.

- Electric Imp is a platform designed to simplify IoT (Internet of Things) development.
- The Electric Imp stands out due to its unique approach of integrating IoT connectivity into a form factor resembling an SD card, making it accessible for various embedded applications.

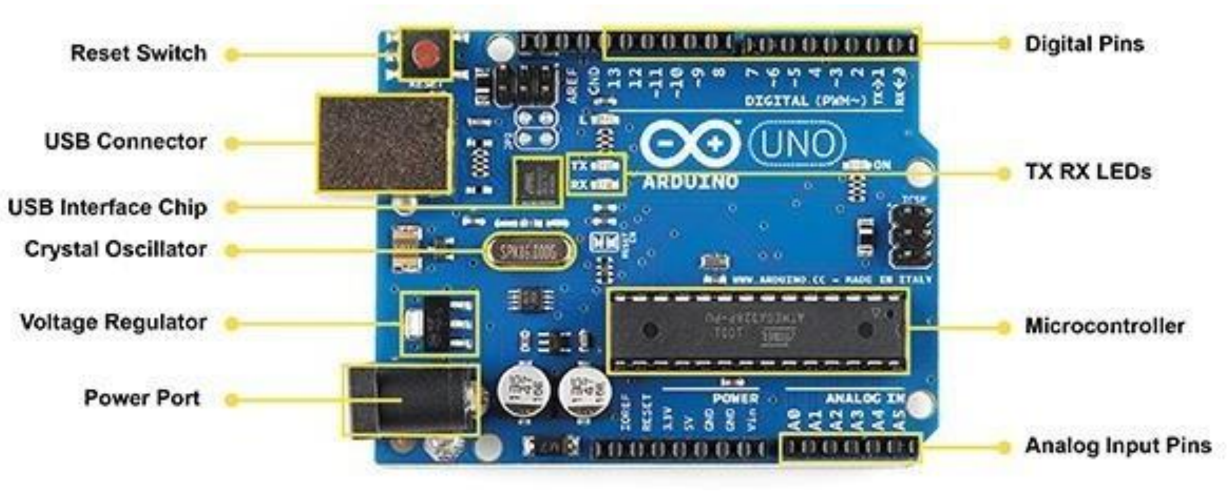
Developing on Electric Imp Platform
1. **Platform Overview**
   - Electric Imp provides a platform for IoT development where you can manage devices and deploy code over-the-air (OTA).

- You interact with devices (Imps) and manage them through the Electric Imp planner application on their website.

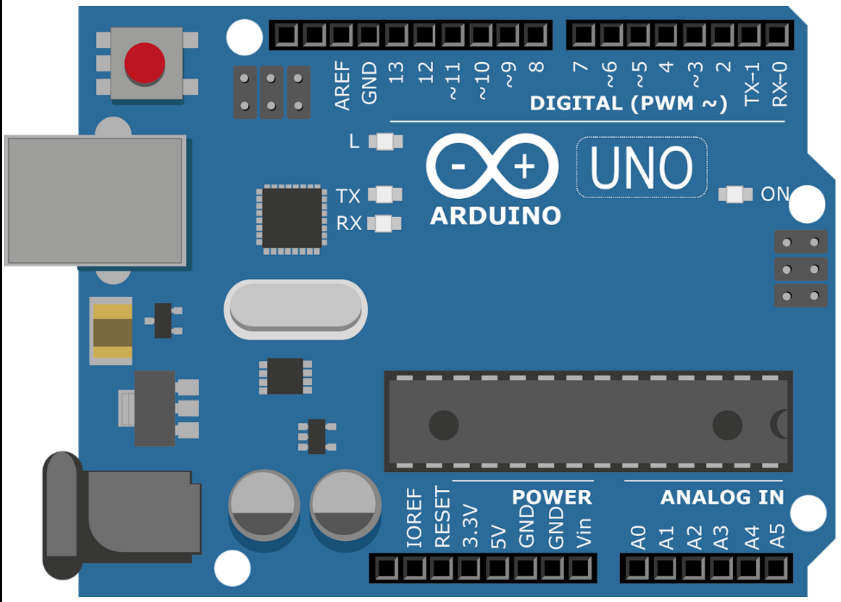
2. **Programming Language: Squirrel**
    - **Description**: Squirrel is the language used for coding on Electric Imp. It's dynamic and has a syntax similar to C, making it flexible for IoT applications.
    - **Documentation and Optimization**: Currently, Squirrel's documentation and optimization are limited compared to more mainstream languages like Lua. This can impact how easily developers find information and optimize their code.
3. **Code Deployment and Execution**
    - **Deployment Process**: After writing code in the Electric Imp IDE, you associate it with specific devices in the planner application.
    - **OTA Updates**: Once associated, devices automatically fetch the latest code updates from Electric Imp's web service when plugged in (impee). This simplifies keeping devices up-to-date without manual intervention.
4. **User Experience**
    - **Platform Interface**: The Electric Imp IDE and planner interface might have a learning curve initially.
    - **Improvements**: As the platform evolves, improvements in documentation, optimization, and usability are expected to enhance developer experience.
5. **Deployment Efficiency and Future Outlook**
    - **Strengths**: Electric Imp excels in streamlined deployment and OTA updates, crucial for efficient IoT device management.
    - **Future Growth**: Continued platform evolution is likely to address current limitations, improving language support, documentation, and user interface for developers.

APPENDIX ARDUINO

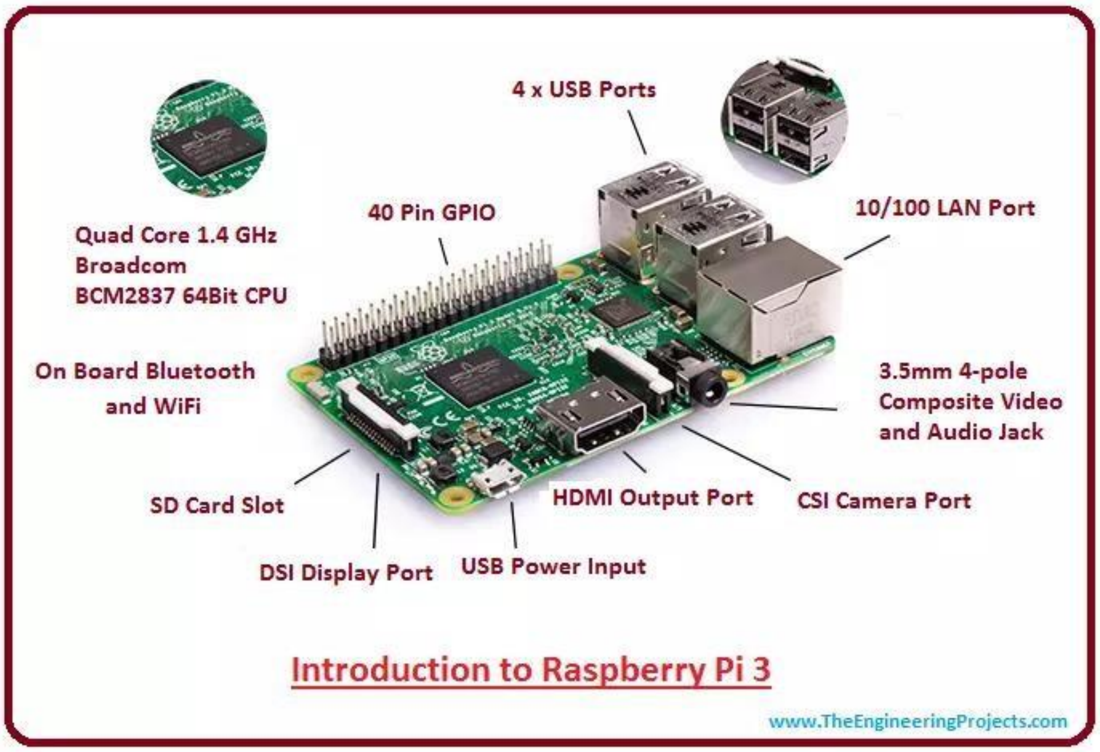
ACTUAL VIEW



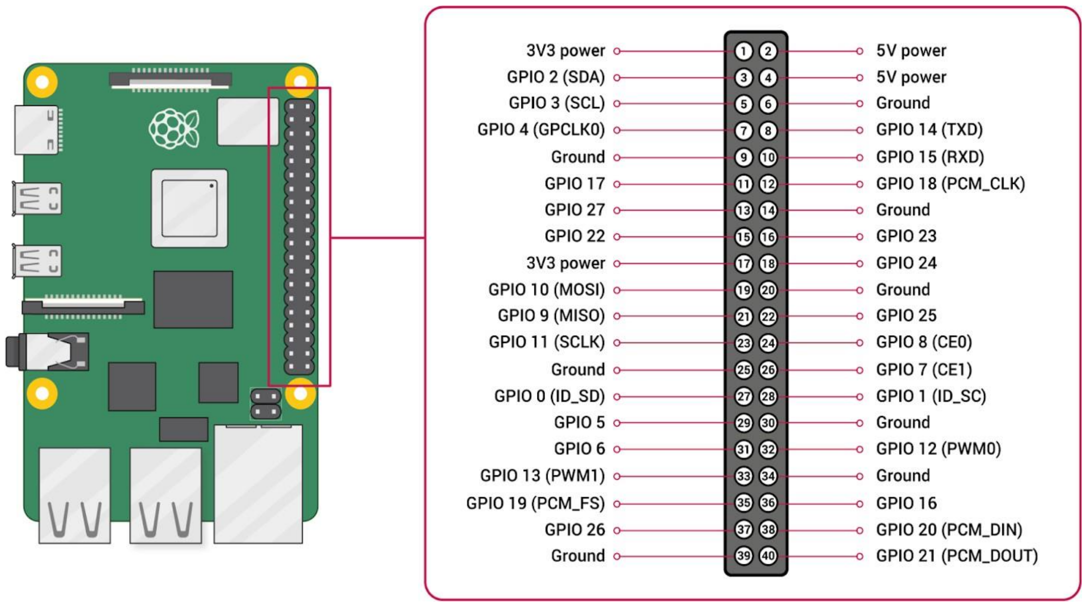
BOARD VIEW



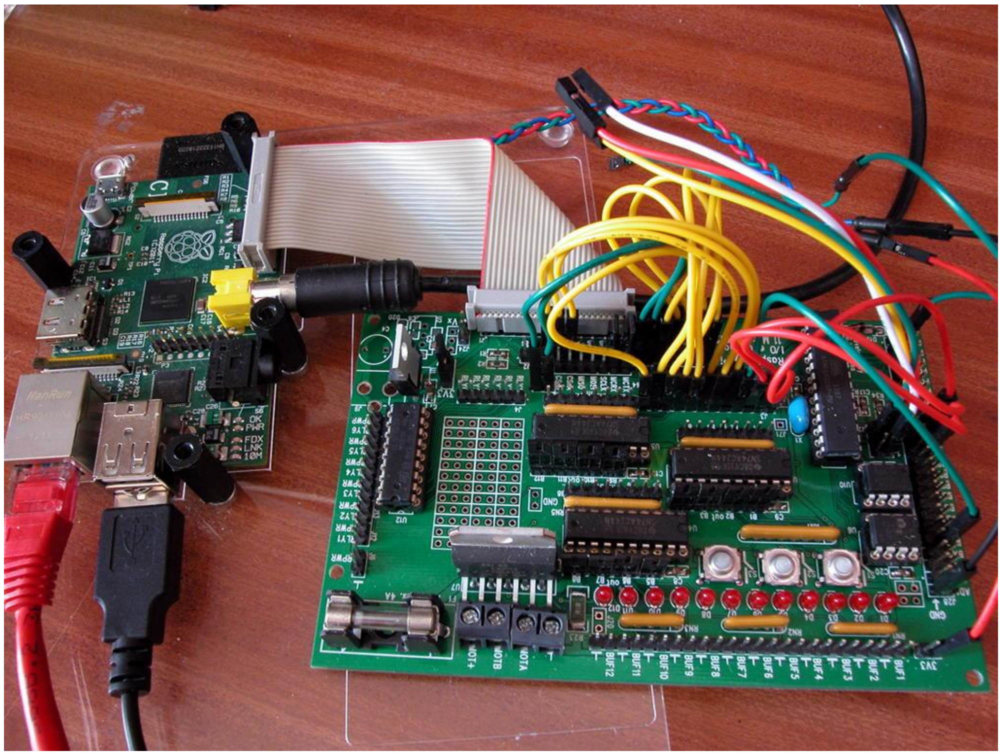
RASPBERRY PI


4 x USB Ports
Quad Core 1.4 GHz Broadcom BCM2837 64Bit CPU
40 Pin GPIO
10/100 LAN Port
On Board Bluetooth and WiFi
3.5mm 4-pole Composite Video and Audio Jack
SD Card Slot
HDMI Output Port
CSI Camera Port
DSI Display Port
USB Power Input
Introduction to Raspberry Pi 3
www.TheEngineeringProjects.com

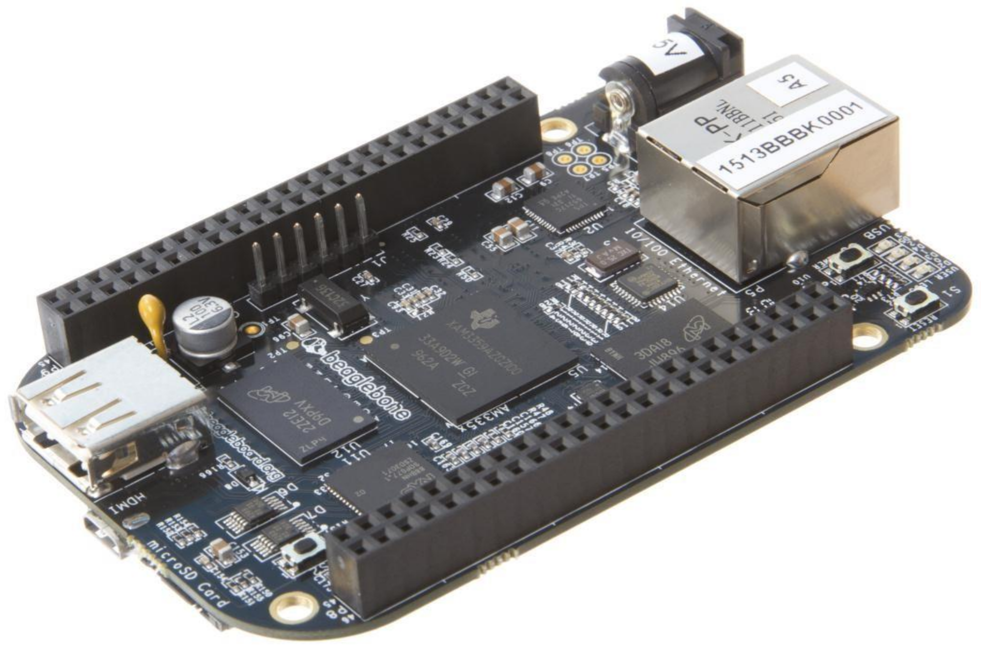
RASPBERRY PI PINS


3V3 power 1 2 5V power
GPIO 2 (SDA) 3 4 5V power
GPIO 3 (SCL) 5 6 Ground
GPIO 4 (GPCLK0) 7 8 GPIO 14 (TXD)
Ground 9 10 GPIO 15 (RXD)
GPIO 17 11 12 GPIO 18 (PCM_CLK)
GPIO 27 13 14 Ground
GPIO 22 15 16 GPIO 23
3V3 power 17 18 GPIO 24
GPIO 10 (MOSI) 19 20 Ground
GPIO 9 (MISO) 21 22 GPIO 25
GPIO 11 (SCLK) 23 24 GPIO 8 (CE0)
Ground 25 26 GPIO 7 (CE1)
GPIO 0 (ID_SD) 27 28 GPIO 1 (ID_SC)
GPIO 5 29 30 Ground
GPIO 6 31 32 GPIO 12 (PWM0)
GPIO 13 (PWM1) 33 34 Ground
GPIO 19 (PCM_FS) 35 36 GPIO 16
GPIO 26 37 38 GPIO 20 (PCM_DIN)
Ground 39 40 GPIO 21 (PCM_DOUT)

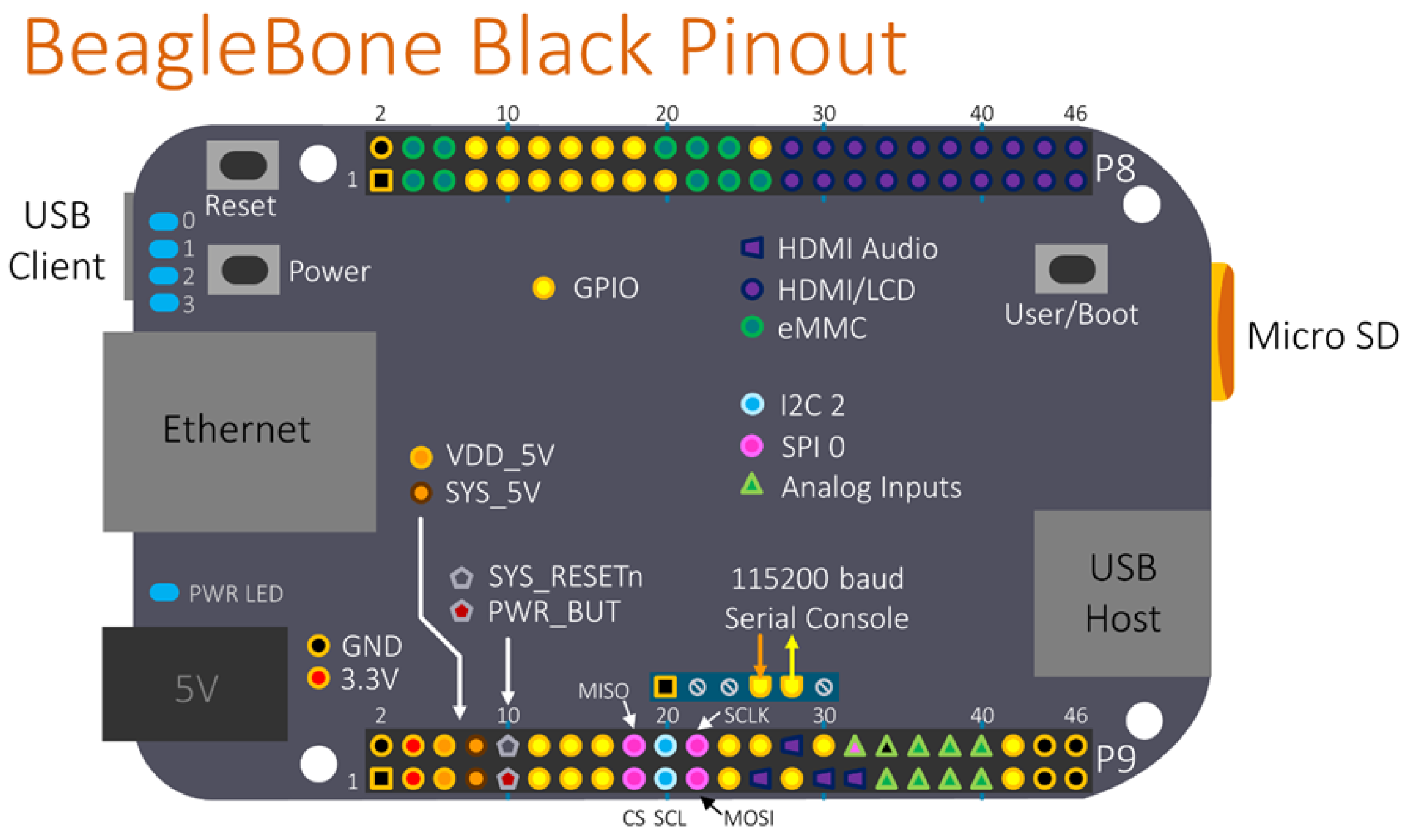
GERTBOARD



BEAGLEBONE BLACK
ACTUAL VIEW



BOARD VIEW


BeagleBone Black Pinout
2
10
20
30
40
46
1
P8
Reset
USB Client
0
1
2
3
Power
GPIO
HDMI Audio
HDMI/LCD
eMMC
User/Boot
Micro SD
Ethernet
I2C 2
SPI 0
Analog Inputs
VDD_5V
SYS_5V
SYS_RESETn
PWR_BUT
115200 baud
Serial Console
USB Host
PWR LED
GND
3.3V
5V
MISO
SCLK
CS SCL
MOSI
P9