Vidyalankar School of Information Technology

Internet of Things

Unit II Exam Question Solution

1 Discuss the trade-offs between cost versus ease of prototyping. (Oct 2018) (May 2023)

- Whenever something is cheap, doing something with it is a bit complex. So, when it comes to programming or any kind of computer or electronic gadget, if its cost is less, the utilization speed and everything will also be less.
- As one increases the cost, the complexity reduces, and it brings ease in prototyping.
- Familiarity with a platform can be important factor considering the relationship between the costs (of prototyping and mass producing) of a platform against the development effort that the platform demands.
- Some of the platform for prototyping are discussed below as per their increasing cost:

i. **Microcontroller**:
  - To create an IOT project, one can ask for an AVR microcontroller chip. Only a chip should be purchased.
  - Need to connect the pins to other components and make a circuit.
  - Need to code this as the code will not be preinstalled so you need to burn on it.

ii. **Arduino**:
  - One can buy Arduino UNO board because the only major advantage is that the labelled headers on the board.
  - It helps to wire up components more easily.
  - Then directly connect it to the system using the USB port, has a well-supported.
  - C++ can be used to increased performance.

iii. **Raspberry Pi:**
  - One can buy the Raspberry Pi or Beagle Bone
  - Install the operating system like linux and run like a mini pc.
  - Writing a code on this is really simple.

iv. **Smartphone:**
  - Stepping upwards, one can buy a smartphone.
  - It has features like: internet connection, input capabilities (touchscreen, button press, camera) and output capabilities (sound, screen display, vibration).
  - Now even in this the coding is really easy when compared to Raspberry Pi .

v. **PC:**
  - If step up more, one can buy a full fledge PC which has internet connection and I/O capabilities.
  - One can write a code at any moment of time and can do anything with it.

2 What are the challenges when we move from prototype to mass production? Explain. (Oct 2018)
Explain the transition from prototype to production. (May 2023)

- **Transition from Prototype to Production:**
- **Changing Embedded Platform:**
  - Transitioning to production often involves considering a change in the embedded platform for reasons like cost and size.
  - Adapting code from a powerful prototype platform to a more restricted, cost-effective one may pose challenges.
  - Compatibility and I/O capabilities of the new platform are crucial considerations.

- **Physical Prototypes and Mass Personalization:**
  - Production techniques used for physical prototypes may require adjustments when scaling to mass production.
  - "Mass personalization" allows for slight customization of each device, potentially commanding a premium.
  - Challenges arise when producing changeable parts in quantities of one while scaling for mass personalization.

- **Climbing into the Cloud:**
  - Transitioning server software from prototype to production is relatively straightforward.
  - Upgrading to more advanced web frameworks, especially when adding user accounts, might be necessary.
  - Business logic often transfers with minimal changes.
  - Scalability in the early stages may involve upgrading to a more powerful server or utilizing cloud computing platforms, offering dynamic scalability based on demand.

3 Discuss open source versus closed source hardware and software. State their advantages and disadvantages. (Oct 2018) (May 2019) (Nov 2022) (May 2023)

- **Open Source:**
  - Open source allows computer source code to be shared and modified free of charge by other users or organisations under a licensing agreement.
  - In case of Open Source, the software is evolved many times and during this evolution, it takes many twists and turns and becomes entirely different than its original version for example, Android.

- **Closed Source:**
  - Closed source is that software that are issued to authorized users or organisations with customized modifications and copying limitations.
  - The software cannot be used or modified by anyone and anywhere. For example, Apple

Merits (Advantages):

- **Coexistence of Open and Closed Source:**
  - Combining open source libraries with closed core business is valuable.
  - Being a "good citizen" by contributing to open source projects, even while using closed source, provides advantages.
- **Flexibility in Project Choices:**
  - Not all work needs to be open source; retaining intellectual property is important for some commercial clients.

- Pragmatically, work may not be polished enough for a viable open release.

- **Varied Licensing:**
- Projects can mix licenses as needed, considering the nature of the work.
- Different components within a project can have different licensing models.

Demerits (Disadvantages):

- **Closed Source for Mass Market Projects:**
- Closed source may be preferred for projects expected to achieve mass market success.
- The advantage of being first to market and the involvement of supply chains are crucial for such projects.
- Protecting IP and configurations may become essential when the project's success attracts potential copycats.
- **Supply Chain Impact:**
- Moving to mass scale can be affected by supply chain considerations.
- The importance of time to market versus competitors influences the choice between open and closed source.

The tension between open and closed source is a significant factor in deciding the most suitable approach for a given project.

4 Explain the following with respect to prototyping embedded devices: Processor speed, RAM, Networking, USB, Power Consumption and Physical size and form factor. (Oct 2018) (Nov 2022) (May 2023)
Discuss the factors we should be considered when deciding to build Internet of Things device. (May 2019)

The following sections discuss about choosing the platform required to design the systems.

- **Processor Speed:** The speed of the processor tells how fast the instruction will be executed. It is always defined in the form of MIPS (Millions of Instructions Per Seconds).
- **RAM:** Random Access memory also known as primary memory is fastest memory available in the system. All the programs are executed in RAM, and it is always better to have higher size of RAM.
- **Networking:** This is the most important thing for IoT, which tells how the devices are connected. It is possible to connect a device with wired LAN but only issue is physical cable. Wireless connection though it gives mobility, but it is a costly affair and is bad in power consumption.
- **USB:** For more powerful computer, tethering to it via USB can be easy way to provide power and networking. So, it is better that microcontrollers include support for USB, so no extra chip is not required.
- **Power Consumption:** Always fast processors consume more power and hence required very high-power supply. For portable device major issue will be power supply.
- **Physical size and form factor.:** With advancements in manufacturing techniques for silicon chips, the size of a chip is now governed by the number of connections the chip needs to make with surrounding components on the printed circuit board (PCB). Surface-mount technology allows for closer packing of chip legs because it doesn't require holes in the PCB for connections.

5	**How is development done for Arduino? Explain. (Oct 2018)**
	• **Development for Arduino:** Arduino development is designed to be simple and accessible, particularly at the prototyping stage. The development process is facilitated by a user-friendly Integrated Development Environment (IDE). This environment is primarily used for writing and compiling code and then uploading it to the Arduino board via a USB cable, which also powers the board. • **Integrated Development Environment (IDE)** The Arduino IDE is straightforward and easy to use. Most projects consist of a single file of code, and the IDE serves as a basic editor with controls for compiling code and pushing it to the board. This simplicity makes it accessible to beginners while still powerful enough for more complex projects. • **Pushing Code to the Arduino** Once the code is written, it is pushed to the Arduino board via a USB connection. This process usually involves selecting the correct serial port and board type. After setting up, the code is compiled and transferred to the Arduino's flash memory. Upon successful transfer, the board reboots and begins running the new code. • **Operating System** By default, Arduino boards do not run a traditional operating system. Instead, they rely on a bootloader that simplifies the code-uploading process. The board runs the uploaded code continuously until it is turned off or reset. For more advanced users, **lightweight real-time operating systems (RTOS) like FreeRTOS** can be uploaded to the Arduino for multitasking purposes. • **Programming Language** Arduino code is written in a **variant of C++**, derived from the Wiring platform. This language is designed to be user-friendly, allowing functions to be called before they are defined, which aids in code readability and maintenance. The code typically includes two main routines: **setup():** Runs once when the board boots, used for initial setup like configuring I/O pins. **loop():** Runs repeatedly while the board is powered on, handling the main logic of the program. • **Debugging** Debugging in Arduino development is primarily done during the compilation stage, where syntax errors or undeclared variables are caught. However, once the code is running on the Arduino, debugging can be more challenging due to the lack of a direct interface for error reporting. Common debugging methods include using serial communication to print messages back to the computer. • **Hardware Interaction** Arduino boards expose General-Purpose Input/Output (GPIO) pins, which are used to interact with various sensors and actuators. These pins are clearly labeled and optimized for easy prototyping. Power for the board and attached components is typically provided through the USB connection during development, although an external power supply can be used for more permanent installations.

6 Compare Raspberry Pi and Arduino. (Oct 2018) (May 2023)

Key Features	**Arduino Due**	**Raspberry Pi Model B**
Purpose	Embedded systems and real-time control	General-purpose computing
CPU Speed	96KB	700 MHz ARM11
RAM	512KB	512MB
OS	Bootloader	Various Linux distributions, other operating systems available
Connections	54 GPIO pins, 12 PWM outputs	65 GPIO pins, of which 8 have PWM
	4 UARTs	1 UART
	Analog and digital pins, fewer built-in connectivity options	Multiple USB ports, Ethernet, HDMI, audio
Programming Languages	Uses simplified C/C++-like programming	Supports multiple languages (Python, C/C++, etc.)
Application Examples	Sensors, actuators, robotics, IoT, automation	Web servers, media centres, programming, robotics, IoT

7 Write note on Sketching. (May 2019) (Nov 2019) (Nov 2022)

- **Sketching in Prototyping**

  Sketching is often the first step when developing a prototype, starting with jotting down ideas or drawing designs on paper. However, the concept of sketching extends beyond just pen and paper to include sketching in hardware and software.
  This process involves exploring the problem space by iterating through various approaches and ideas to determine what works and what doesn't. The focus is on the ease and speed of trying things out rather than the fidelity of the prototype.

  For physical design, sketching might involve using materials like LEGO, foamcore, or cardboard to quickly prototype and test ideas. This allows for rapid experimentation and iteration without getting bogged down by the details.

  In the context of electronics and software, sketching can involve building quick prototypes to test specific functionalities or concepts.
  An example of this is the work done during the Little Printer hackday, where participants quickly developed and tested ideas using minimal code and readily available hardware like an Arduino Ethernet board.
  The emphasis was on getting a working prototype that could be demonstrated, even if it wasn't polished or fully functional.

  Sketching in this broader sense allows developers to explore and refine their ideas before committing to a more detailed and time-consuming development process.
  It is a vital part of the prototyping stage, enabling rapid feedback and iterative improvement.

8	**Write Short note on sensor and actuators.**
	• **Sensors and Actuators in Embedded Devices** • Sensors and actuators are crucial components in embedded devices, facilitating interaction with the external environment. • **Sensors** are input devices that gather information about the surroundings and feed this data into the system. For example, a light-dependent resistor (LDR) measures ambient light level. It adjusts the system's response based on the amount of light detected, making it useful in applications like automatic lighting systems. Another example is a thermistor, which measures temperature and can be used in climate control systems, ensuring the environment remains within desired temperature ranges. - **Temperature Sensor (DHT11):** Measures temperature and humidity, converting environmental data into digital signals, widely used in smart homes and environmental monitoring. - **Ultrasonic Sensor (HC-SR04):** Uses high-frequency sound waves to detect object distances, commonly applied in robotics and proximity detection. • **Actuators**, on the other hand, are output devices that allow the system to perform actions in the physical world. For instance, LEDs can be used to provide visual feedback by lighting up in different colors, while DC motors can drive mechanical movements, such as rotating a fan or moving a robotic arm. • The combination of sensors and actuators enables embedded systems to sense changes in their environment and respond accordingly, making them essential for applications in automation, robotics, and the Internet of Things (IoT). - **Servo Motor:** Provides precise control over angular or linear motion, extensively used in robotics, automation, and drone control. - **Solenoid:** Converts electrical energy into linear motion, often utilized in locking mechanisms and valve control systems.
9	**Write short note on Raspberry Pi. (May 2019) (Nov 2022)**
	• The Raspberry Pi is a low-cost, credit card-sized computer created for educational purposes. • It features a Broadcom BCM2835 system-on-chip, with a powerful GPU capable of high-definition video and a 700 MHz ARM CPU. • The Raspberry Pi, unlike the Arduino, is a fully capable computer that can run modern operating systems like Linux, interact with peripherals such as keyboards and monitors, and connect to the internet via built-in Ethernet or Wi-Fi dongles. • It is widely used in education and hobbyist projects for programming, media centers, and IoT applications. • **Operating System** - The Raspberry Pi can run various operating systems, but the most popular choice is **Raspbian**, a Debian-based Linux distribution. - Other options include Adafruit's **Occidentalis**, which is optimized for headless use (without a keyboard and monitor) and enables remote access by default through SSH.

- **Programming Language**
  - **Python** is the recommended programming language for Raspberry Pi, making it ideal for educational programming.
  - It offers ease of use but comes with trade-offs in terms of speed and memory management compared to lower-level languages like C++.

- **Debugging**
  - Debugging on the Raspberry Pi is more advanced compared to the Arduino.
  - It offers tools like Python's integrated debugger, Linux's **strace** command, and system logs, allowing developers to troubleshoot issues while the system is running.
  - Python's error-handling features also help in logging errors and managing memory leaks that could occur in long-running IoT projects.

- **Some notes on the hardware**
  - The Raspberry Pi offers **8 GPIO pins**, but they are 3.3V tolerant, meaning care must be taken when interfacing with 5V devices to avoid damage.
  - It also lacks analog inputs, so external ADCs are needed to connect analog sensors.
  - Powering the Pi through USB can be inconsistent, making it important to use reliable power sources.

- **Openness**
  - Most of the Raspberry Pi's components and software, including custom Linux distributions like Raspbian, are open-source.
  - However, the **Broadcom BCM2835 chip is proprietary**, and its full datasheet is not available, though drivers and tools are actively developed by Broadcom employees and released as open-source software.

10 **"Open source has a competitive advantage". Discuss. (Nov 2019)**

- While open source is often viewed as a gift economy or a community-driven effort of good citizens, it also serves as a powerful competitive advantage.

- **Low-Risk and Cost-Effective**:
  - Open source software is tested, debugged, and improved by many contributors, ensuring a level of robustness and reliability.
  - Businesses can use open source software in both open and closed-source projects, as long as the licensing permits.
  - For instance, rather than building everything from scratch (like microcontrollers, libraries, HTTP stacks, etc.), companies can leverage pre-built open-source solutions like Arduino, Python, or Ruby on Rails.
  - This can significantly reduce development costs and avoid vendor lock-in compared to commercial solutions.

- **Mindshare and Community Support**:
  - Aggressively using open source can help a product gain mindshare quickly.
  - Products like Arduino have become popular not solely because of their superior performance but because of their open designs, which allow for community contributions and the development of compatible hardware (e.g., Arduino shields).
  - This open approach attracts a wide user base that contributes to product improvement, making the product more appealing to developers and the broader tech community.

- **Openness as a Marketing Strategy:**
  - By exposing interfaces like a Linux shell or open protocols, open-source projects build goodwill among the "geek" community.
  - This community often champions the product because it isn't a proprietary "black box."
  - Their enthusiasm and endorsement can turn a well-designed open-source project into a widely-adopted platform.

11 How can one tap into the community for promoting IOT devices? Explain. (Nov 2019) (Nov 2022)

- To promote IoT devices effectively, tapping into the community of makers, hobbyists, and developers can be a powerful strategy. The community offers not just a customer base but also a network for collaboration, mentorship, and innovation.

- **Access to Knowledge and Problem Solving:**
  - By engaging with established platforms with large communities, such as Arduino, you gain access to a wealth of knowledge.
  - If you face a technical problem or need guidance, the community has likely encountered similar issues.
  - A simple search can lead you to blog posts, videos, or code snippets that solve your problem, allowing faster development and troubleshooting.

- **Mindshare and Scalability:**
  - As your IoT project grows, having the support of a large and active community can help it gain mindshare.
  - This is essential when scaling up your operations, especially when you need to recruit skilled individuals who are already familiar with your chosen platform.
  - For example, Arduino has a significant mindshare, making it easier to find people with the necessary skills.

- **Mentorship for Beginners:**
  - For newcomers to IoT, community support is invaluable. Local maker meetups, hackspaces, and workshops like Maker Night Liverpool offer face-to-face mentorship, which can help accelerate learning.
  - Such gatherings allow beginners to work with experienced makers who can guide them through basic projects and help troubleshoot problems.
  - This mentorship fosters a collaborative spirit, which is key in the IoT space.

- **Local and Online Communities:**
  - Engaging with both local and online communities enables you to discuss your IoT projects in a supportive environment.
  - Sharing ideas with others allows you to receive feedback, improve your designs, and build connections.
  - While the anonymity of online platforms can sometimes lead to unsupportive behaviour, face-to-face meetings at hackspaces tend to be more encouraging, making them an excellent entry point for new makers.

- **Creativity and Innovation:**
  - The IoT community is at the frontier of innovation, often compared to the early days of website creation (like Geocities).
  - Though some projects may seem experimental or rough, this creative environment is where future groundbreaking IoT devices and solutions will emerge.

	- By engaging with this community, you place yourself at the cutting edge of IoT development, surrounded by the creativity that can fuel the next generation of successful products.
12	**With the help of an example explain the process of Scaling up the electronics. (Nov 2019) (May 2023)**
	• Scaling up electronics is the process where a circuit is built from initial testing through prototype to a finished PCB. • The starting point for prototyping is usually a Breadboard. This lets u push-fit components and wires to make up circuits without requiring any soldering and therefore makes experimentation easy. • When we are happy with how things are wired up its easy to solder the components, which may be sufficient to make the circuit more permanent and prevent wires from getting away. • **Example:** Let us consider an evolution of the part of Bubblino circuit, from initial testing through prototype, to finished PCB. - The first step in creating a circuit is generally to build it up on a breadboard, this way we can easily reconfigure things as we decide exactly how it should be laid out. - When we are happy with how the circuit works, soldering it onto a stripboard will make the layout permanent. This means you can stop worrying about any wire getting loose, and if we are going to make only one copy of the circuit, then it might be as far as we need to take things. - If we need to make copies of the circuit, or if we want a professional finish, we can turn our circuit into a PCB, this makes it easier to build up the circuit because the position of each component will be labelled, there will be holes only where the components go, and there will be less chance of short circuits because the track between components will be protected by the solder resist. - When you want to scale things even further moving to combined board allows us to remove any unnecessary components from the microcontroller board
13	**Explain the following IOT devices built with Arduino.** **(i) The Good Night Lamp (ii) Botanicals (iii) Baker Treat (Nov 2019)**
	(i) The Good Night Lamp: The Good Night Lamp is a connected lamp system designed to foster a sense of connection between people, even when they are far apart. It consists of a "big lamp" and one or more "little lamps" that are paired together. When the big lamp is switched on or off, the little lamps, which can be placed anywhere in the world, reflect the same state. This allows individuals to see when their loved ones are turning their lamps on or off, creating an ambient connection to their daily routines. The product was built using Arduino due to its simplicity, cost-effectiveness, and the ability to customize it for mass production. One key challenge the team faced was ensuring easy internet connectivity for non-technical users, which led to the exploration of Wi-Fi and GSM/3G connectivity options. **(ii) Botanicals:** Botanicals is an innovative IoT device that helps bridge communication between plants and humans. It monitors the moisture level of

the soil in plant pots and notifies the owner when the soil gets too dry. This is humorously described as "interspecies communication," where the natural signals of plants, such as leaf colour or drooping, are translated into human notifications through mediums like telephone, email, or Twitter. Botanicals uses Arduino-compatible microcontrollers and allows the plants to essentially "call" their owners when they need water.

(iii) Baker Treat: BakerTweet is a physical device that allows bakers to send tweets about freshly baked goods directly from the bakery. Designed to withstand the tough environment of a bakery, where flour, dough, and heat are prevalent, BakerTweet communicates via WiFi and is housed in a robust, bakery-proof box. It allows bakers to notify their customers of freshly made products without needing a computer or phone. Built using Arduino, an Ethernet Shield, and a WiFi adapter, BakerTweet provides a simple, easy-to-use interface that allows bakers to send messages to Twitter without the hassle of handling sensitive electronics in a challenging environment.

14 Explain microcontrollers and system-on-chips with respect to embedded computing. (Nov 2022)

- Microcontrollers and system-on-chips (SoCs) are fundamental elements of embedded computing, especially in IoT devices. Here's an explanation of each in the context of embedded computing:

- **Microcontrollers:**

- Microcontrollers are the "brains" behind countless IoT sensors and automated systems.
- They are highly specialized, single-chip devices that integrate a processor, RAM, and storage into a compact package, allowing them to perform specific tasks efficiently. Unlike desktop computers with separate components for processing, memory, and storage, microcontrollers combine all these elements into a single, compact unit, making them ideal for devices that need to be small, efficient, and focused on a particular function.
- Most microcontrollers still use 8-bit architecture. However, these modern microcontrollers are much faster, smaller, and more power-efficient than their 1980s counterparts.
- While limited in computing power, with RAM often measured in kilobytes and storage in the tens of kilobytes, they are highly effective for specific tasks like controlling sensors or machinery.
- A popular example of a microcontroller in IoT is the **Arduino platform**, which uses Atmel's AVR ATmega microcontroller. Arduino's simplicity and GPIO pins make it easy to interface with various sensors and motors.

- **System-on-Chips (SoCs):**

- System-on-chips (SoCs) are a more powerful alternative to microcontrollers, sitting between microcontrollers and full-fledged PCs. SoCs also integrate a processor and peripherals onto a single chip but offer significantly more processing power, often ranging from a few hundred megahertz to several gigahertz.
- Unlike microcontrollers, SoCs typically require more RAM, often measured in megabytes, and can run more complex applications.
- SoCs also rely on external storage solutions, such as SD cards, to expand their capabilities.

	- Due to their greater power and complexity, SoCs require an operating system to manage resources, with embedded versions of Linux and Microsoft systems being popular choices. - Examples of SoCs include the **Raspberry Pi** and **BeagleBone**, which are popular in IoT projects that require more processing power, greater flexibility, and the ability to run a full operating system. - Both microcontrollers and SoCs are critical to the development of IoT devices, but they are used in different contexts depending on the device's needs for computing power and functionality.