Session 8: MicroProcessors & Micro Controllers

Overview

- Select a microcontroller based on a set of requirements;
- Specify and select communication protocols dependent on a set of requirements;
- Understand microcontroller to microcontroller communication;
- Understand microcontroller to computer/Cloud communication;

Computer

- A key component for developing an IoT solution is a **computer** that reads the sensors, stores and processes data, communicates and displays results, or actuates some other device
- There are literally thousands of different types, with a vast array of capabilities, costs and features, so this raises questions, such as:
  - Which micro controller/microcomputer should we use?
  - How do we get them to talk to the Internet and ‘Cloud’?
  - What’s the difference between a microcontroller and microcomputer?

Microcontrollers and Microprocessors

- Almost all electronics equipment contains either a **microcontroller** or **microprocessor**

Microcontrollers

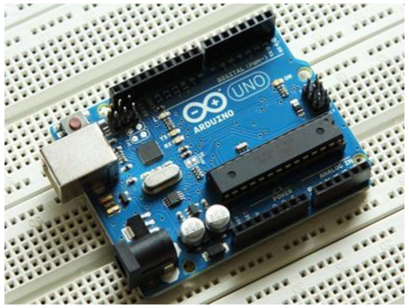
- Most IoT systems utilize **microcontroller** units (**MCUs**) to collect data and transfer it to the processing units through the Internet. The advantages of using **MCUs** include:
  - They are relatively simple, as **MCUs** do not require an operating system to function and are easy to interface with external components such as sensors.
  - A **MCU** is usually sufficient to provide processing power and functionality in most IoT systems, thus making them economically viable for IoT applications.
  - Due to their simplicity, MCUs are less vulnerable to security attacks.

  Example a **digital watch**: it contains a small ***microcontroller*** that continually loops and counts time (seconds, minutes, and hours progression). **Washing machines** contain a **microcontroller** that controls the operation of the machine; from the on/off switch, to activating the different phases of the washing machine: soak, wash, spin, dry.

Microprocessor

- A **microprocessor** is a micro version of our computers, hence the name ‘microprocessor’.
- We can see that generally, if an application requires a form of Operating System, a microprocessor is normally used. A good example is our smartphones, which often use ARM microprocessors.

Microcontrollers vs microprocessors

	Microcontroller	Microprocessor
Components	Processor, RAM, IO are all on the one chip	No RAM, ROM and IO pins
	Controlling bus is internal and cannot be changed	Interfaces to peripherals so board is expandable
Examples	Ardiuno Uno *Img src: Image by roelofse* 	ARM processor *Img src: By Socram8888 [CC BY 2.0], from Wikimedia Commons*

Microcontroller or microprocessor?

- Under general circumstances, one of the main constraints of an IoT microcontroller or microprocessor is their power consumption. However, there are other key aspects to consider when selecting a micro. They include:
  - **Power Consumption**: Most IoT devices will be placed in the field running on a small battery. We would like the devices to run for weeks, days or even years on a single battery.
  - **Connectivity Options**: For example; Bluetooth, LoRa, 802.11 Choosing the IoT connectivity comes with its own set of choices, dependent mostly on power and range required.

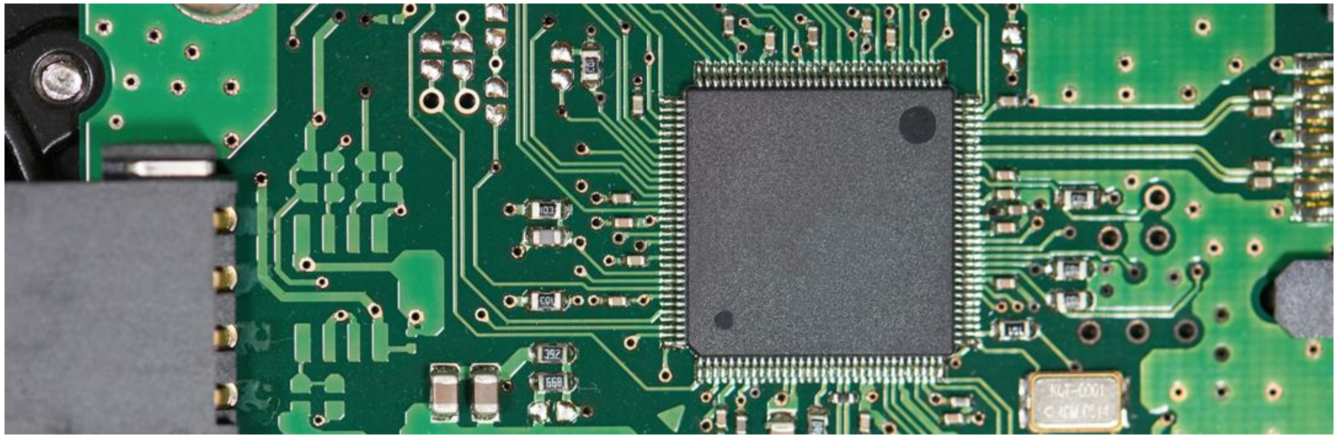
Microcontroller or microprocessor?

- **Development boards and kits:**
  - Many IoT development boards are prototyping solutions that feature low-power processors and support various programming environments such as ***'C', Python, Scratch***, etc. These platforms are generally used to collect sensor data using firmware and transfer it to an on-site or cloud-based server.
  - There are more powerful hardware prototyping platforms based on microcomputers, such as the Raspberry Pi and BeagleBoard. Such microcopmuters are enhanced with multiple core processors, may run an operating system, and support multiple types of peripherals.

High-level capabilities

- We can characterize IoT devices in terms of these high-level capabilities, including:
  - Power management
  - Sensors and actuators
  - Processing capability
  - Connectivity

Prototyping boards

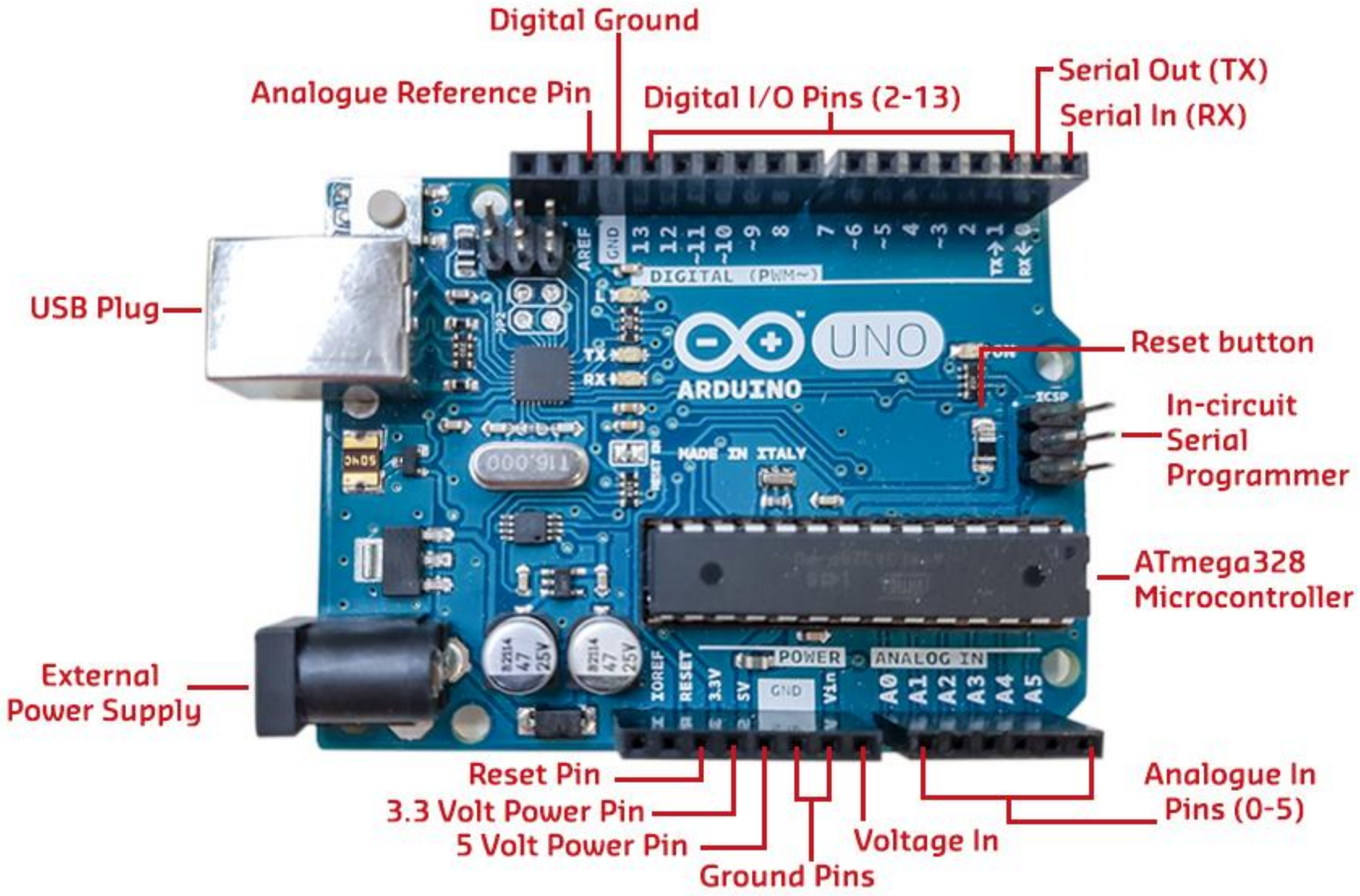


- The are a growing range of off-the-shelf prototyping kits now available, many specifically designed for IoT development. These low-cost, modular systems offer a flexible way to rapidly develop and test out IoT solutions without a big financial outlay. Many of these microcontrollers and microcomputers (also called single board computers, or SBCs), are designed around System-on-a-Chip (SoC) ICs.

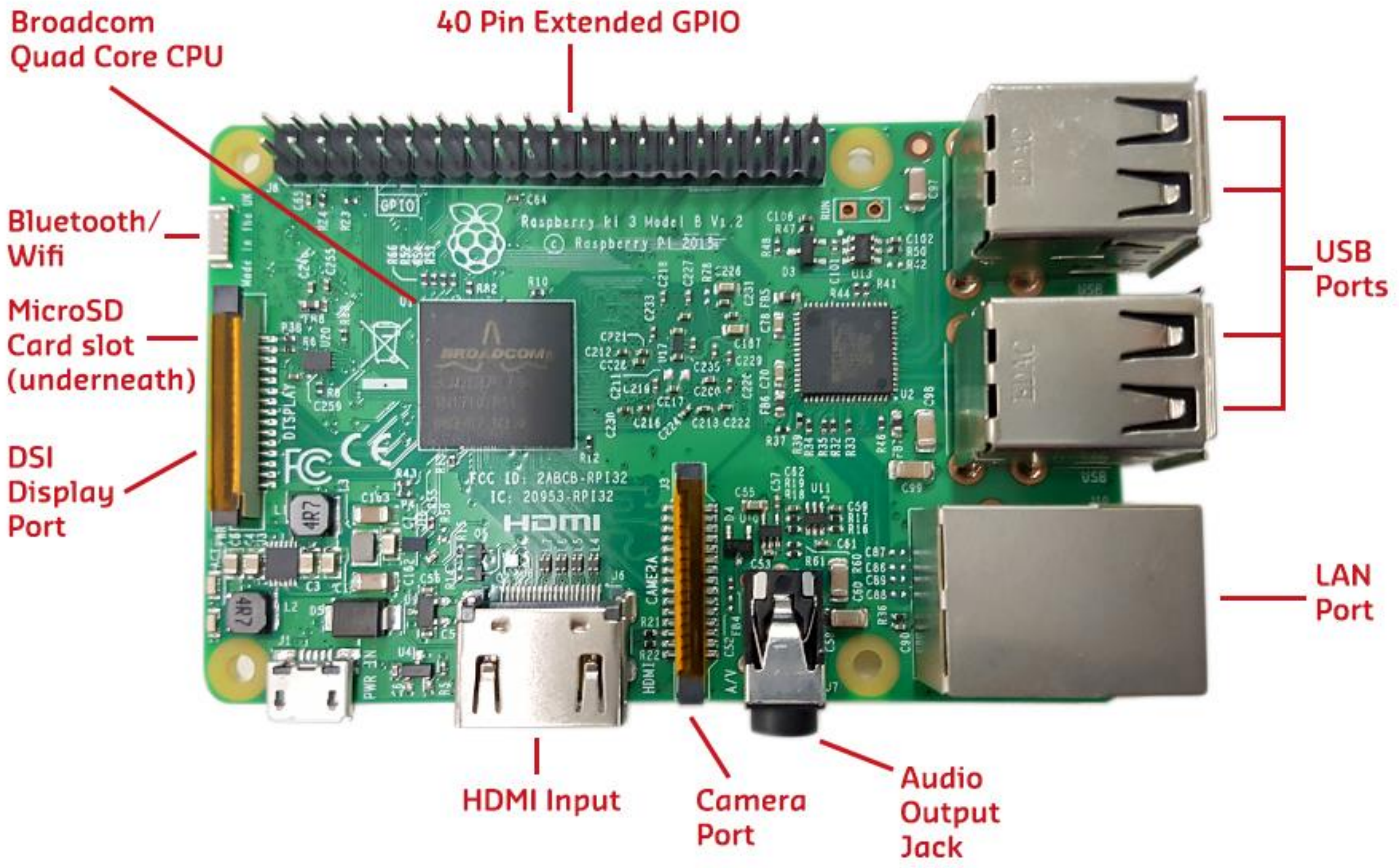
Electronic Toolkits

- **Arduino** is an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board. You can develop interactive objects that take input from a variety of switches or sensors to control lights, motors, and other physical objects. Go to Arduino at http://arduino.cc.
- While the Arduino is not suitable for use as a computer, its low power requirement makes it capable of controlling other devices efficiently.
- **The Raspberry Pi** is a low cost, credit-card-sized computer that plugs into a computer monitor or TV. You operate it using a standard keyboard and mouse. It is capable of doing everything a computer can do, from browsing the Internet and playing high-definition video, to making spreadsheets, word-processing, and playing games. Go to Raspberry Pi at http://www.raspberrypi.org.
- **The Beaglebone** is very similar to the Raspberry Pi in size, power requirements, and application. The Beaglebone has more processing power than the Raspberry Pi; therefore, it is a better choice for applications with higher processing requirements. Go to Beaglebone at http://beagleboard.org.

Microcontroller development (*Arduino Uno*)



MicroComputer development boards (*Raspberry Pi 3, Model B)*



Selecting a micro

Feature	Arduino Nano	Raspberry Pi 3B+
Processor	ATmega328P 16MHz AVR	Broadcom, BCM2837B0 1.4GHz 64 bit ARMv8 Cortex A53
Memory	32kB flash, 2kB RAM	1GB SDRAM
GPIO	14 (inc 6 PWM)	40
Analogue Pins	8	
Network (BT/Wifi)	-	Gigabit Ethernet, WiFi (802.11b/g/n/ac), Bluetooth 4.2, BLE
Runs an OS	No	Raspbian, Ubuntu, Mate, Windows, IoT core…etc (many more 3rd party options)
Power Usage	19mA at 5V 95mW	350mA at 5V (idle) 1.75W (5W max)
Cost	USD 22	USD35

Overview of common languages for IoT

- **Arduinos** have great libraries and support for the C language.
- **Raspberry Pi** are well suited to Python (but as they run Linux, they support many languages).
  - The **C language** is powerful, efficient and very popular in constrained devices (limited memory/processing power) but is rather cryptic and difficult to use well. C Works at a 'low level' meaning it interacts well with the hardware and I/O.
  - **Java** is another popular language, and is cross platform It has a version specifically for embedded systems (IoT): Java SE Embedded.
  - **Python** is a scripting language, which unlike C does not require compiling. It is the preferred language for use on the Raspberry Pi, and is easy to learn with support from a large, helpful community.