Lesson 3 – Wireless comm. technologies

- S.P. Chong

Objectives

- In this lesson, you will learn the basics of wireless communication technologies: **frequency channels**, **data rates**, **coverages** & **applications** of common wireless comm. techniques such as **Bluetooth**, **WiFi** & **Cellular** (3G/4G).
- You will also learn basic **modulation techniques**, such as **ASK, FSK, PSK**.
- You will also learn basic **multiple access techniques**, such as **polling, contention, time & frequency division multiple access**.
- For other wireless comm. technologies such as **ZigBee, Lora, Sigfox, RFID, NFC & GPS**, you only need to be able to list some applications. BTW, in the previous lesson, you have learnt to use RFID for **identification** & GPS for **localization**.

Common wireless technologies

- In 1895, Guglielmo **Marconi** succeeded in sending wireless signals over a distance of 1.5 miles.
- In 1896, he demonstrated it in London and obtained the first patent in wireless telegraphy.
- In 1909, he was awarded the Nobel Prize for Physics for his discovery of radio waves.
- With his discovery, a new industry emerged.

Common wireless technologies (cont.)

- Wireless technology is a term used to describe telecommunication in which **electromagnetic waves** carry signal over part or all of the communication path.
- Due to different **throughput** requirements for different **applications** (**voice, data, video**) and different **coverage** (**PAN, LAN, WAN**), different wireless technologies are being used.
- It is important to select the right wireless technology for different IoT applications.

Common wireless technologies (cont.)

- Many wireless communication technologies are in use today.
- Do you know how some of these are used?

Common wireless technologies (cont.)

Wireless LAN (Local Area Network).

Wireless WAN (Wide Area Network).

Wireless PAN (Personal Area Network).

Access control, theft prevention, tracking.

Cashless payment.

Localisation.

LPWAN (Low Power Wide Area Network).

Common wireless technologies (cont.)

https://en.wikipedia.org/wiki/Bluetooth

https://en.wikipedia.org/wiki/Wi-Fi

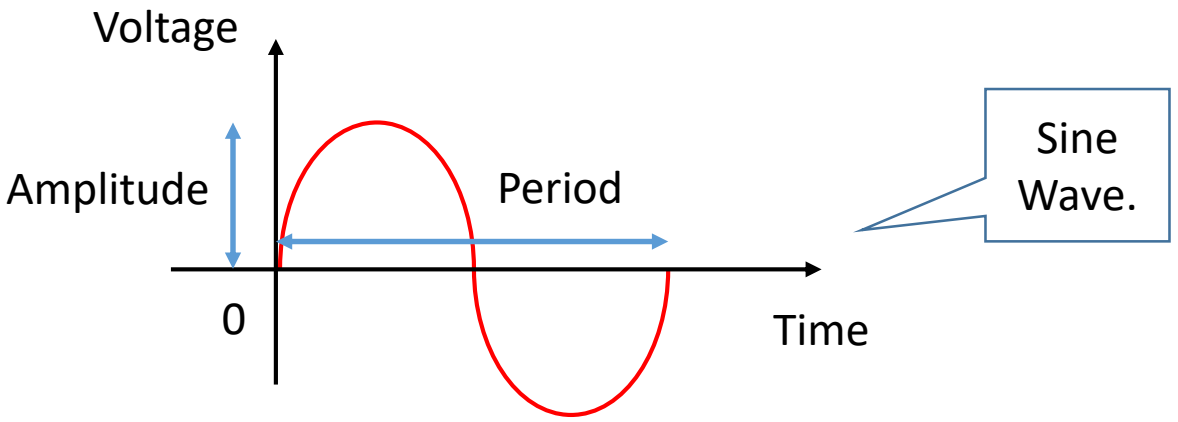
https://en.wikipedia.org/wiki/4G

- How do these technologies vary in terms of **frequency** used, **data rate**, **coverage**, and **applications**?

	Frequency	Data Rate (peak)	Coverage	Applications
Bluetooth	2.4 GHz	3 Mbps	Wireless PAN ~10 M	Connecting mobile phone to headset, or PC to speakers. Transferring file from mobile phone to PC.
WiFi	2.4 GHz or 5 GHz	Few 100 Mbps	Wireless LAN ~20 M	Home or office network – connecting mobile phone or PC to AP / router for internet access.
Cellular (4G LTE)	1.8 GHz & 2.6 GHz (Singtel)	100 Mbps on cars or trains, 1 Gbps for pedestrians or when stationary	Wireless WAN (via base stations)	Voice, data, video over long distance.

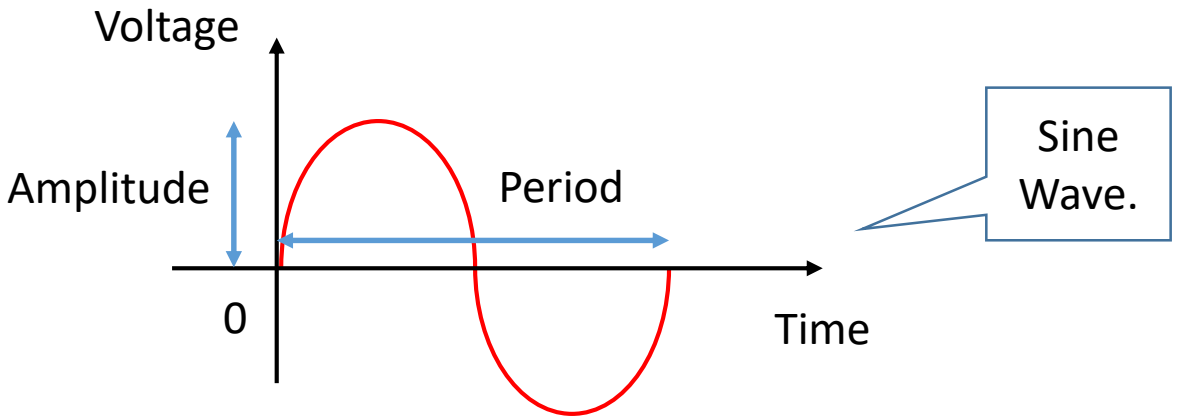
Basic Modulation Techniques

- **Digital data**, in the form of **binary numbers** 10011101_2, are used in mobile phones, tablets, laptops, smart watches etc.
- How can these be sent "**over the air**" i.e. wirelessly?
- Answer: the digital data is first "**modulated**", or converted into **analogue form**, before transmission.
- Basic modulation techniques include **ASK** (Amplitude Shift Keying), **FSK** (Frequency Shift Keying) and **PSK** (Phase Shift Keying).



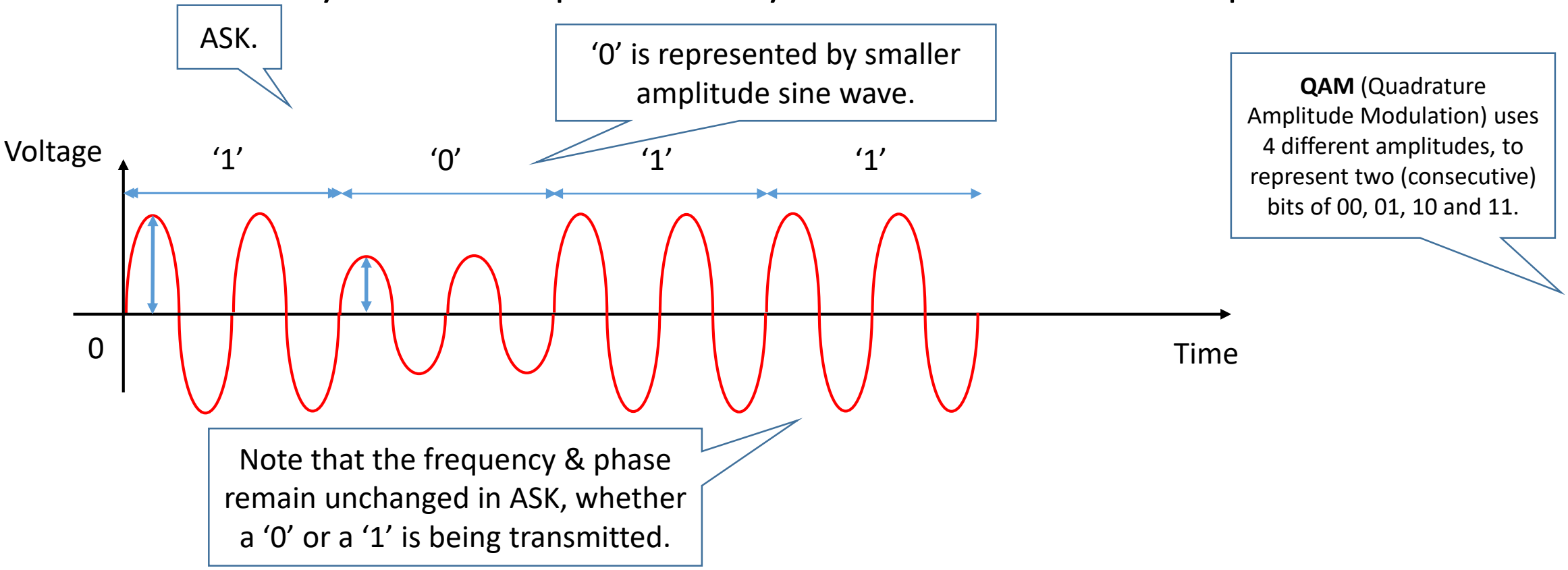
Basic Modulation Techniques (cont.)

- A **sinusoidal wave**, such as a **sine wave**, is characterised by 3 parameters: **Amplitude, Frequency** and **Phase**.
- **Frequency = 1 / Period**. For instance, if a sine wave has a period of 2 seconds, its frequency is ½ Hertz.
- What is Phase?
- Phase refers to which part of the sine wave appears at a reference point e.g. the **origin** (0 in the figure below). If the positive zero crossing part appears at the origin, the phase is said to be 0 degree.



Basic Modulation Techniques (cont.)

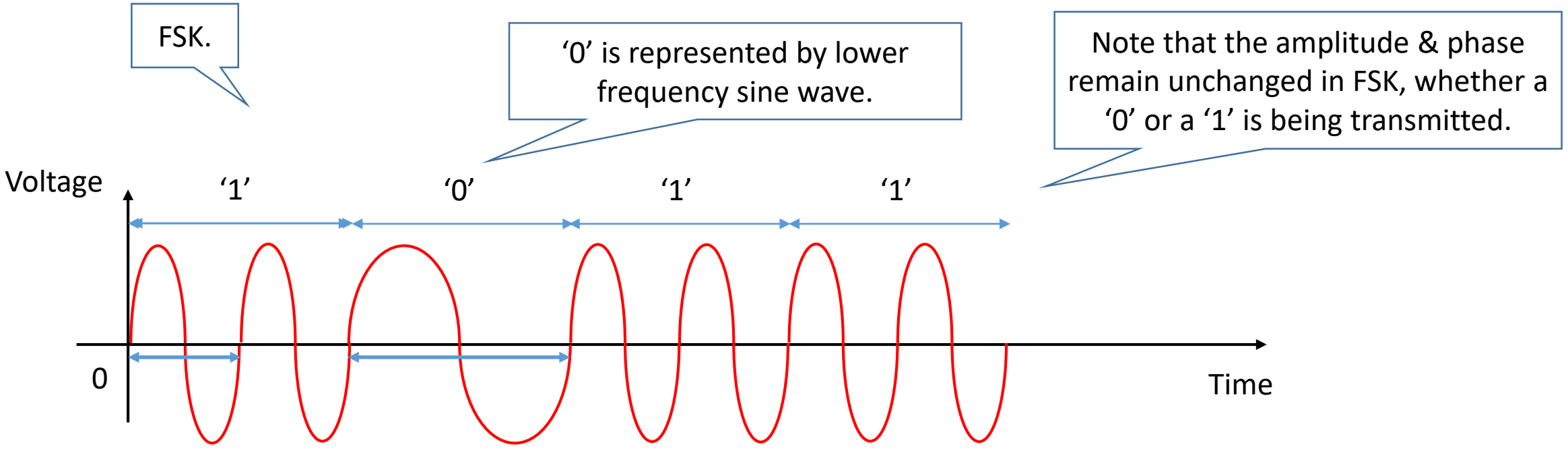
- What is **Amplitude Shift Keying**?
- This means a binary bit of 1 is represented by sine wave of one amplitude, while a binary bit of 0 is represented by sine wave of another amplitude:



QAM (Quadrature Amplitude Modulation) uses 4 different amplitudes, to represent two (consecutive) bits of 00, 01, 10 and 11.

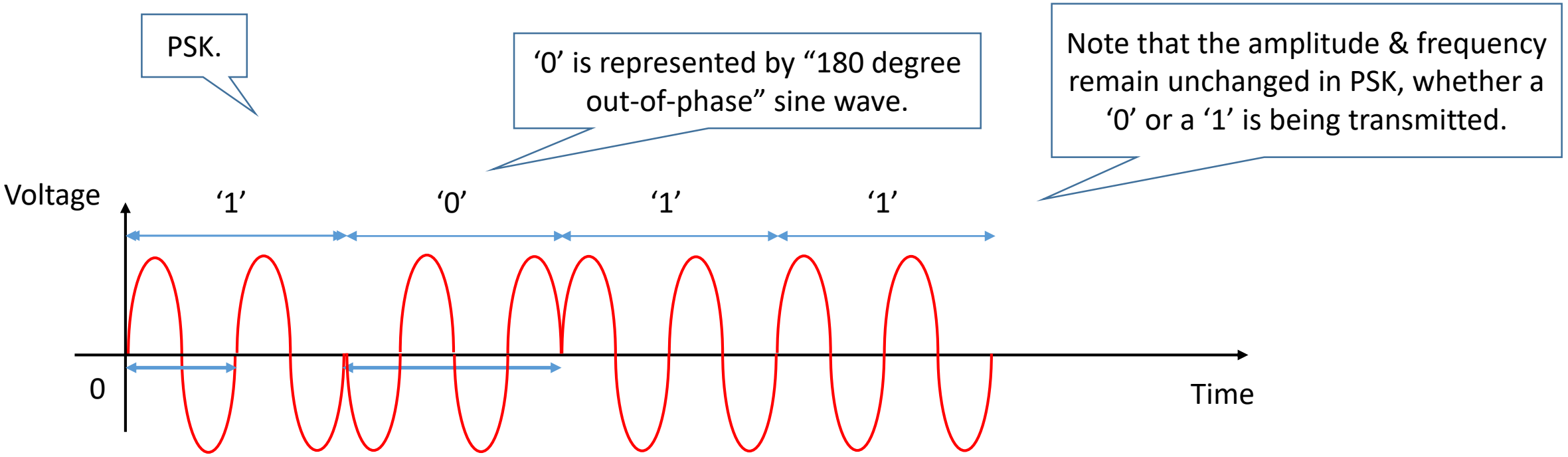
Basic Modulation Techniques (cont.)

- What is **Frequecy Shift Keying**?
- This means a binary bit of 1 is represented by sine wave of one frequency, while a binary bit of 0 is represented by sine wave of another frequency:



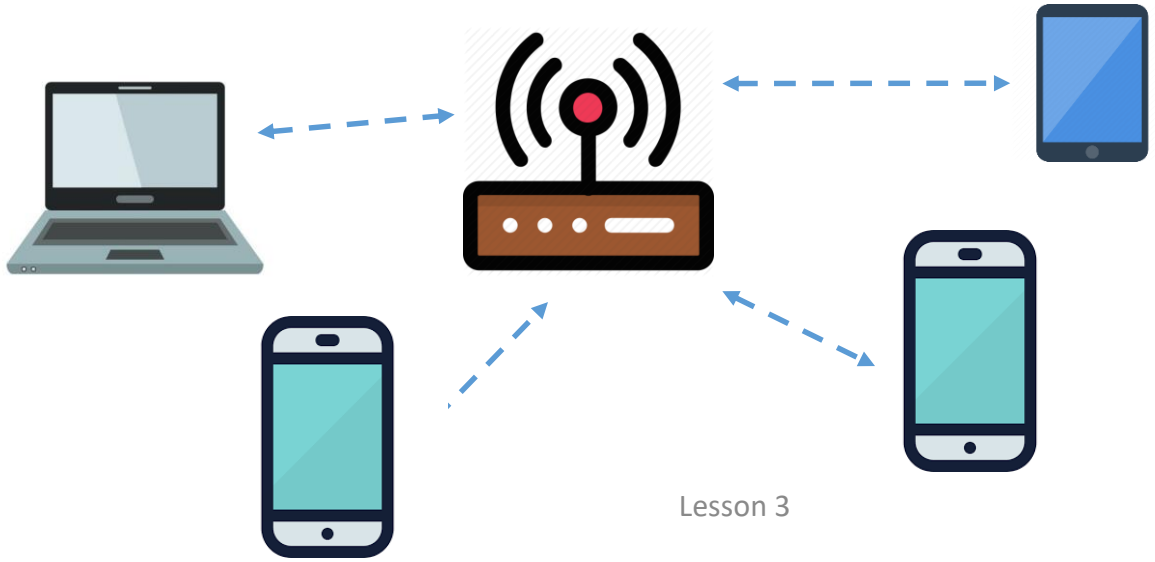
Basic Modulation Techniques (cont.)

- What is **Phase Shift Keying**?
- This means a binary bit of 1 is represented by sinusoidal wave of one phase, while a binary bit of 0 is represented by sinusoidal wave of another phase:



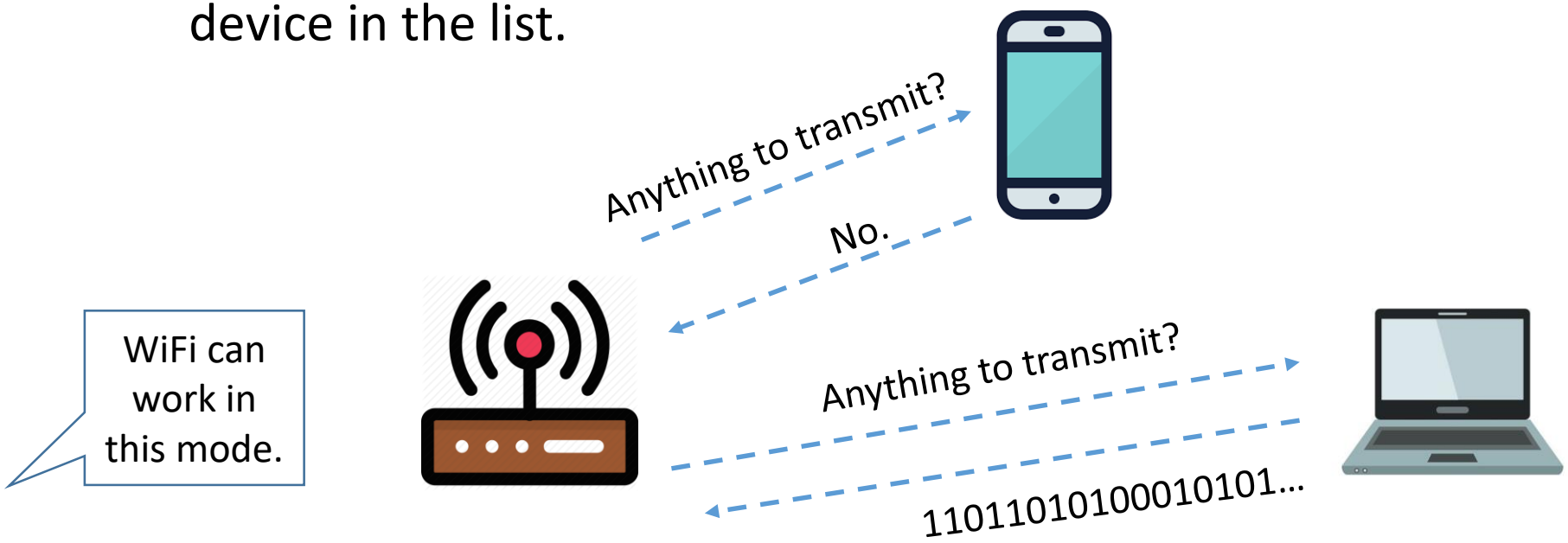
Basic Multiple Access Techniques

- Multiple access is a mean to allow **multiple communication streams** to happen over time, or at the same time, using the same wireless communication technique (such as Bluetooth, WiFi or Cellular).
- For instance, how can many laptops, tablets and mobile phones connect to the same WiFi AP (Access Point) using the same frequency (2.4GHz)?
- This is a complicated topic, so we will only touch briefly on the meaning of 1. polling, 2. contention, 3. frequency division multiplexing and 4. time division multiplexing.



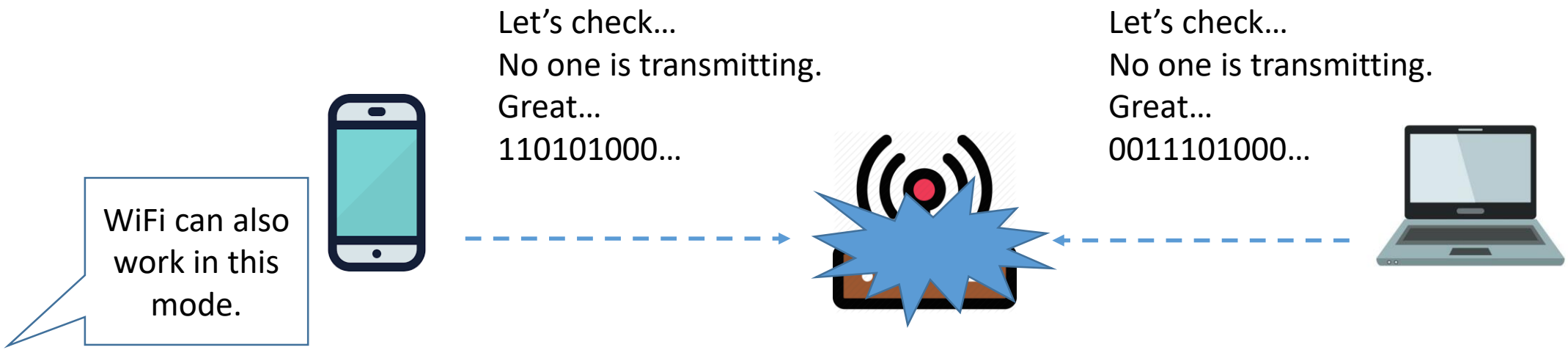
Basic Multiple Access Techniques (cont.)

- When "**polling**" is used, a device such as a router, can be made a **Coordinator**.
- The Coordinator will ask/**poll** each device in turn whether it has anything to transmit.
- If the device "polled" has data to transmit, it will do so, for a certain duration of time.
- If the device "polled" has nothing to transmit, the Coordinator will ask/poll the next device in the list.



Basic Multiple Access Techniques (cont.)

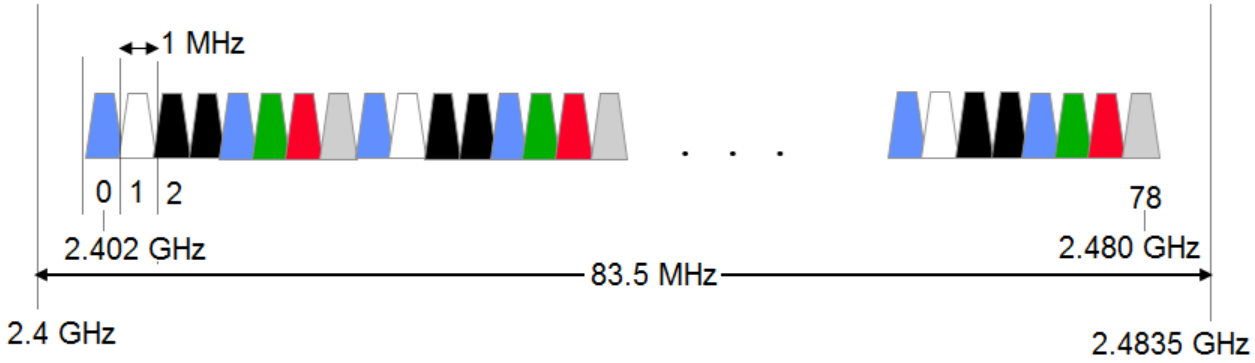
- When "**contention**" is used, a device (say A) that has data to transmit, first **senses** if anyone else is transmitting.
- If someone is transmitting, A will try again later. If no one is transmitting, A will transmit.
- There is a possibility that another device (say B) also senses no one transmitting. So it transmits at the same time as A, and a **collision** occurs.
- If collision occurs, each device will wait a random amount of time before **re-transmitting**.



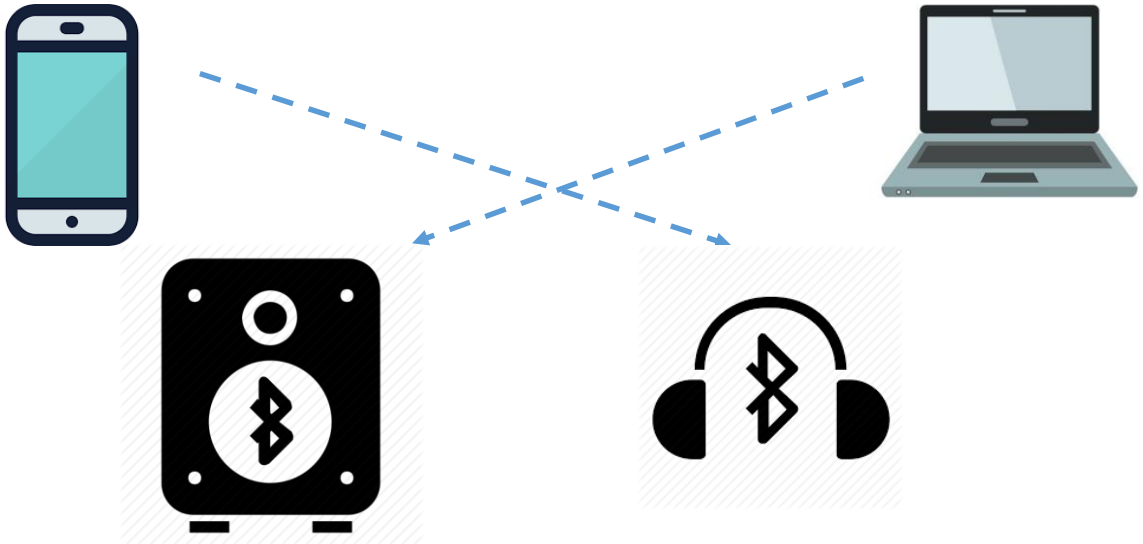
Basic Multiple Access Techniques (cont.)

- When "**frequency division multiplexing**" is used, a pair of devices may use a **frequency channel** different from another pair of devices.

Bluetooth uses this, more exactly FHSS (Frequency Hopping Spread Spectrum) i.e. a pair of devices keeps changing the frequency channel used.



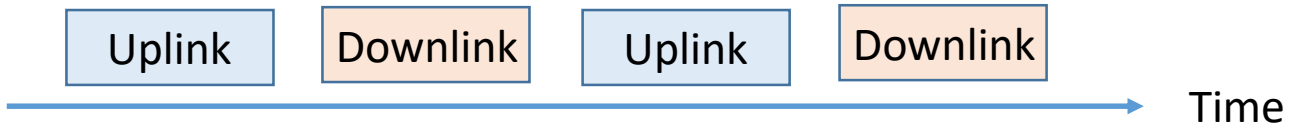
I use BT channel 1 centred at 2.403GHz.



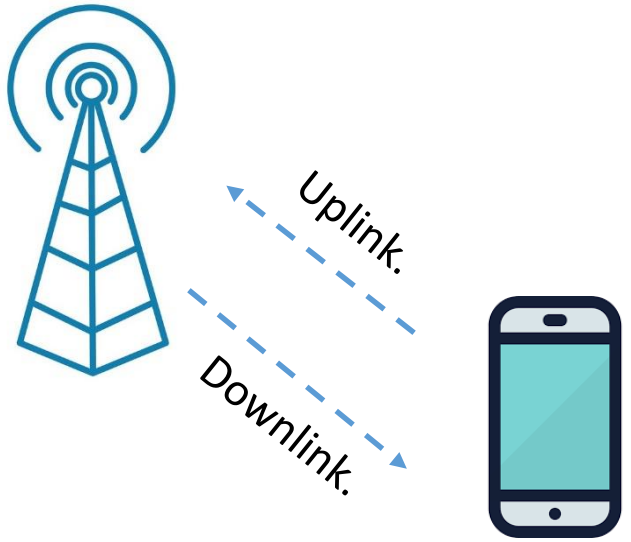
I use BT channel 3 centred at 2.405GHz.

Basic Multiple Access Techniques (cont.)

- When "**time division multiplexing**" is used, a few pairs of devices share the same frequency channel **over time**.



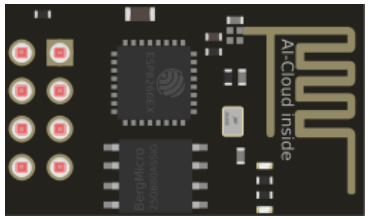
Cellular (LTE, 4G) uses this (and also frequency division)



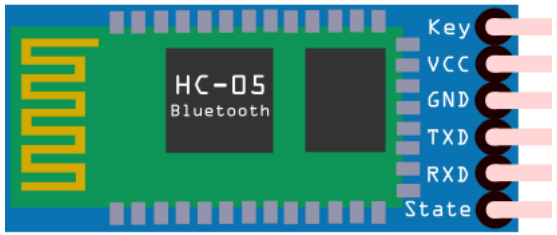
Bluetooth module & Wifi module

- In the lab exercises, you will learn to use the Bluetooth module HC05 & the Wifi module ESP01.
- Each can be connected to an UNO (**soft**) **serial** port.
- You will learn to use "**AT commands**" to **configure** these modules, and to **transmit** data wirelessly.

ESP01 Wifi module.



HC05 Bluetooth module.



Lab Exercises

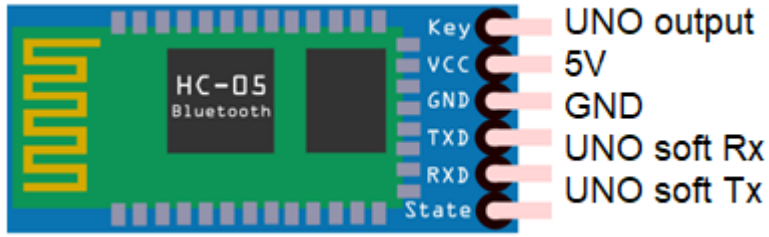
- Exercise 3.1 – Using BT module
- Exercise 3.2 – Using WiFi module – connections
- Exercise 3.3 – Using WiFi module – basic AT commands
- Exercise 3.4 – Using WiFi module – uploading sensor data to the cloud

Exercise 3.1 – Using BT module

Intro

- If the **Key** (or **EN**) pin is **HIGH**, HC05 operates in the "**AT command mode**" – AT commands can be issued to change settings.
- E.g., AT+NAME="BT123" will changes the name to BT123.
- By default, all HC05 are named as just that, "HC05", which makes it difficult to differentiate between different devices, when you scan to make a connection.
- If the Key (or EN) pin is **LOW**, HC05 operates in the "**communication mode**" – data can be transmitted wirelessly between this and another Bluetooth device, such as your mobile phone.

Connection



Sample Code

See next page…

Applications

- Wireless communication in a PAN (Personal Area Network) e.g. remote control & monitoring

Exercise 3.1 – Using BT module (cont.)

HC05_AT_command_mode | Arduino 1.8.2

File Edit Sketch Tools Help

HC05_AT_command_mode §

```
// HC05 (Bluetooth module) sample code (AT command mode)

#include <SoftwareSerial.h>

SoftwareSerial mySerial(3, 2); // RX, TX

void setup() {
  Serial.begin(9600);
  while(!Serial);
  Serial.println("Enter AT command & click Send...");
  mySerial.begin(9600);

  pinMode(7, OUTPUT);
  digitalWrite(7, HIGH); // EN = HIGH to enter "AT command mode"
}

void loop() {
  if (mySerial.available()) {
    Serial.write(mySerial.read());
  }
  if (Serial.available()) {
    mySerial.write(Serial.read());
  }
}
```

Connect HC05's TXD to UNO's pin 3.

Connect HC05's RXD to UNO's pin 2.

Connect HC05's EN to UNO's pin 7.

This allows AT commands to be issued, to change settings.

When the program is run, open the Serial Monitor. Send AT+NAME="your name" to give the HC05 a unique name.

Sample Code / Run

AT command mode

COM7 (Arduino/Genuino Uno)

AT+NAME="ChongSP"

```
Enter AT command & click Send...
OK
```

OK will appear when you Click send.

Exercise 3.1 – Using BT module (cont.)

HC05_communication_mode | Arduino 1.8.2

File Edit Sketch Tools Help

HC05_communication_mode

```
// HC05 (Bluetooth module) sample code (communication mode)

#include <SoftwareSerial.h>

SoftwareSerial mySerial(3, 2); // RX, TX

void setup() {
  Serial.begin(9600);
  while(!Serial);
  Serial.println("HC05 in \"communication mode\"...");
  mySerial.begin(9600);

  pinMode(7, OUTPUT);
  digitalWrite(7, LOW); // EN = LOW to enter "communication mode"
}

void loop() {
  if (mySerial.available()) {
    Serial.write(mySerial.read());
  }
  if (Serial.available()) {
    mySerial.write(Serial.read());
  }
}
```

Sample Code / Run

Communication mode

This allows wireless communication over a short distance.

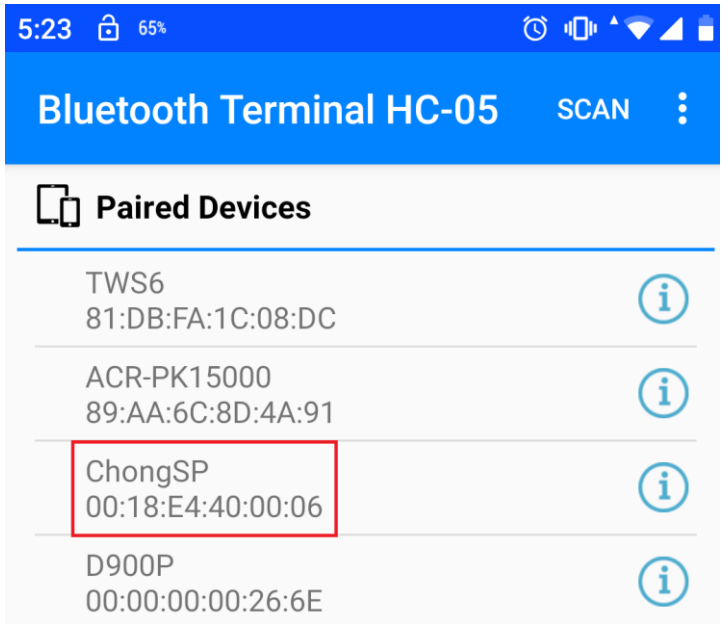
When the program is run, open the Serial Monitor. Use a mobile app (e.g. Bluetooth Terminal HC-05) to send something to the UNO.

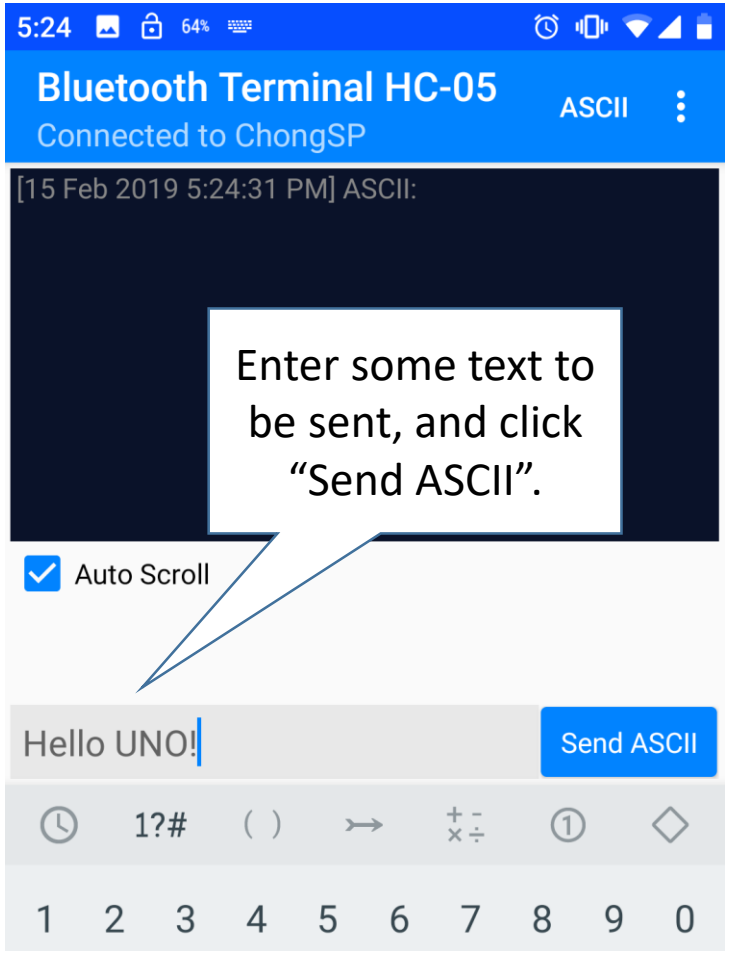
Exercise 3.1 – Using BT module (cont.)

Bluetooth Terminal HC-05
mightyIT
4.3★
INSTALLED

This is the mobile app to be installed. (There are other alternatives.)

You should be able to find the HC05 named as “your name”.

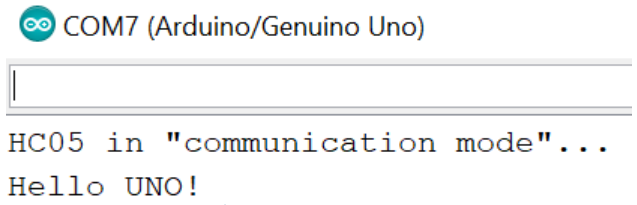




Enter some text to be sent, and click “Send ASCII”.

Sample Code / Run

Communication mode



The text will appear in the Serial Monitor.

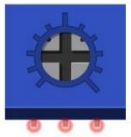
You can thus use HC05 in a remote control application, for instance, a “1” received could mean turning on the LED.

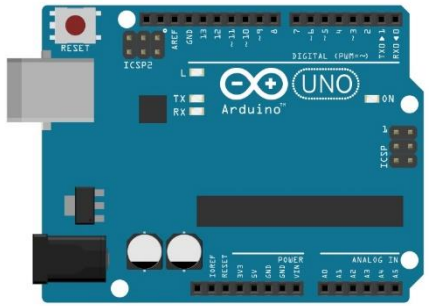
Exercise 3.2 – Using WiFi module – connections

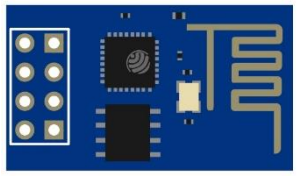
Intro

- In Exercises 3.2 – 3.4, you will:
  - use a shield to connect a potentiometer & an ESP01 (UART - WiFi module) to an UNO,
  - program the UNO to read from the potentiometer,
  - send the readings via the ESP01 to a cloud platform called Thingspeak.
- You will learn more about Thingspeak platform in the next lesson.

Sensor (potentiometer) → Microcontroller (Arduino UNO) → WIFI module (ESP8266-01)



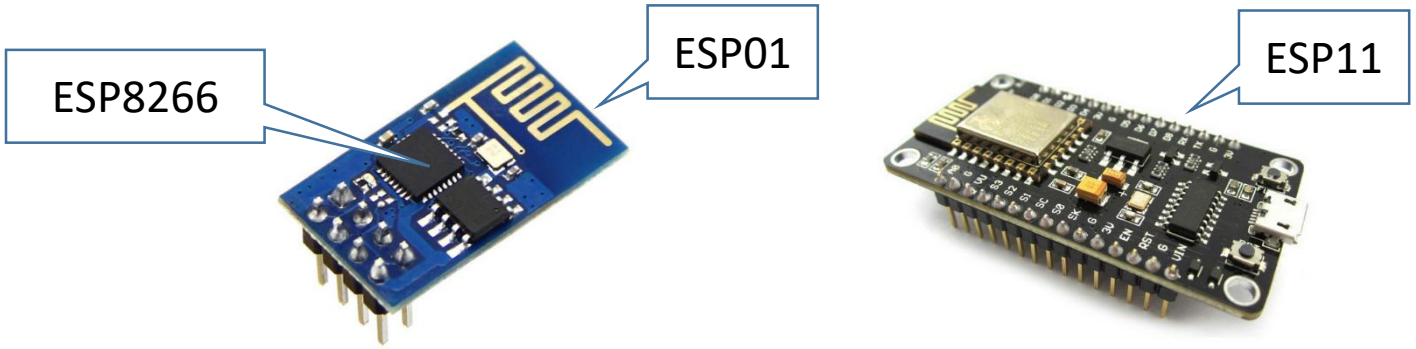




Cloud (*thingspeak.com*)

Exercise 3.2 – Using WiFi module – connections (cont.)

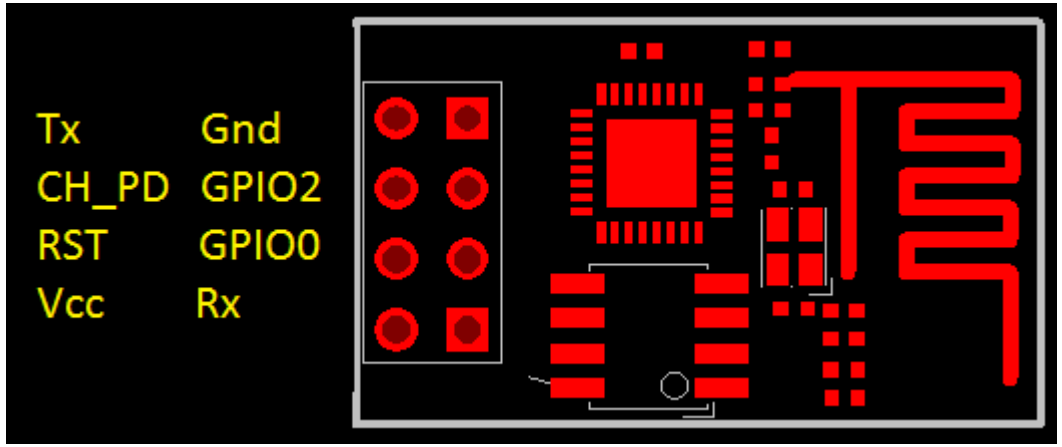
Nice to know



- According to **Wikipedia**, the **ESP8266** is a **low-cost Wi-Fi chip** with full **TCP/IP stack** and **microcontroller** capability produced by Shanghai-based Chinese manufacturer, Espressif. **ESP01** (the **board**) is produced by AI-Thinker.

Connection

ESP01 uses 3.3V. So “level shifting” is needed to connect to 5V UNO.



Exercise 3.2 – Using WiFi module – connections (cont.)

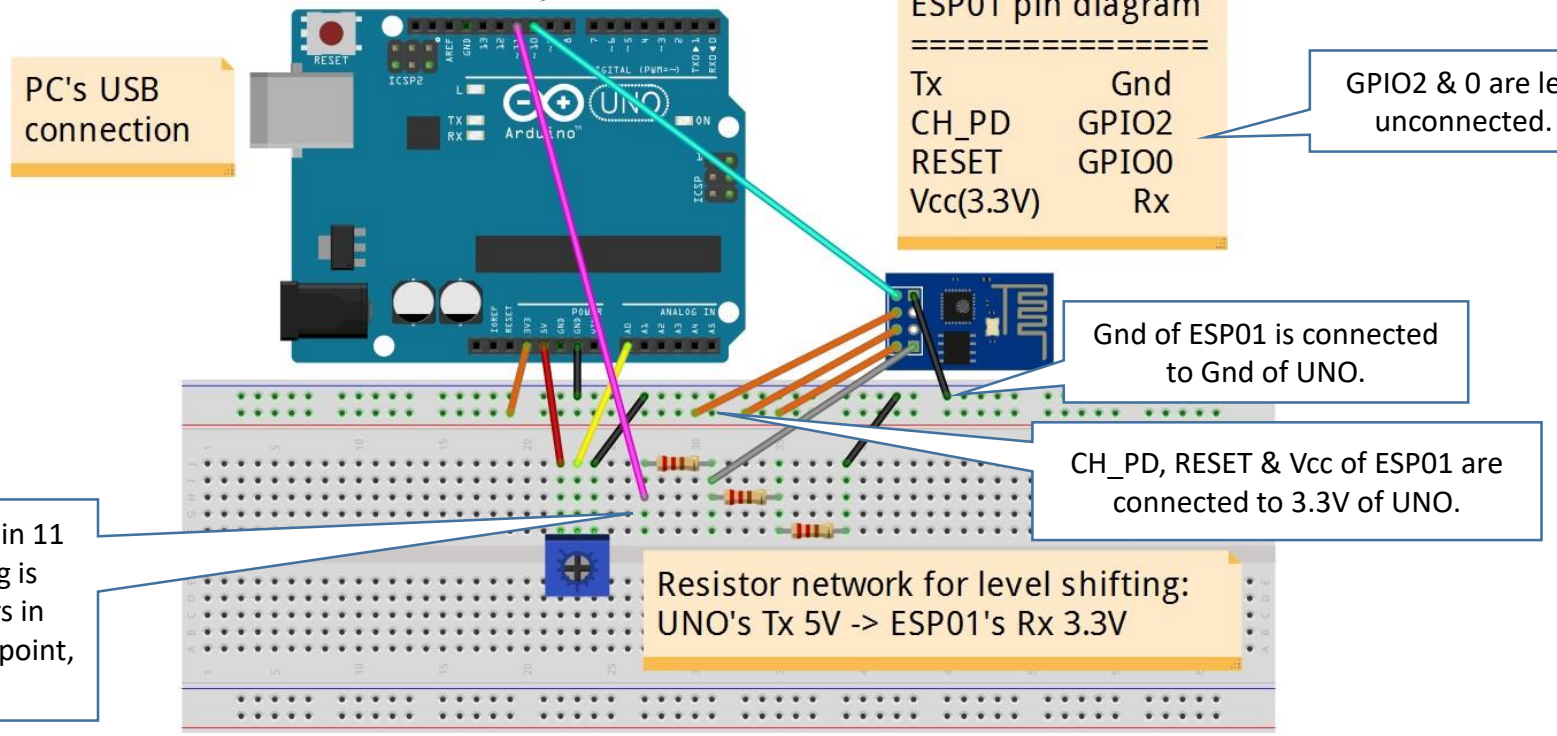
Connection (cont.)

By default, UNO serial port uses pins 1 (Tx) & 0 (Rx). This is already used for UNO – PC connection.

So, a software serial port (pin 11 for Tx & pin 10 for Rx) is used for UNO – ESP01 connection.

Tx of ESP01 is connected to pin 10 i.e. Rx of UNO. No level shifting is necessary as a HIGH output from the 3.3V ESP01 is still treated as a HIGH in the 5V UNO.

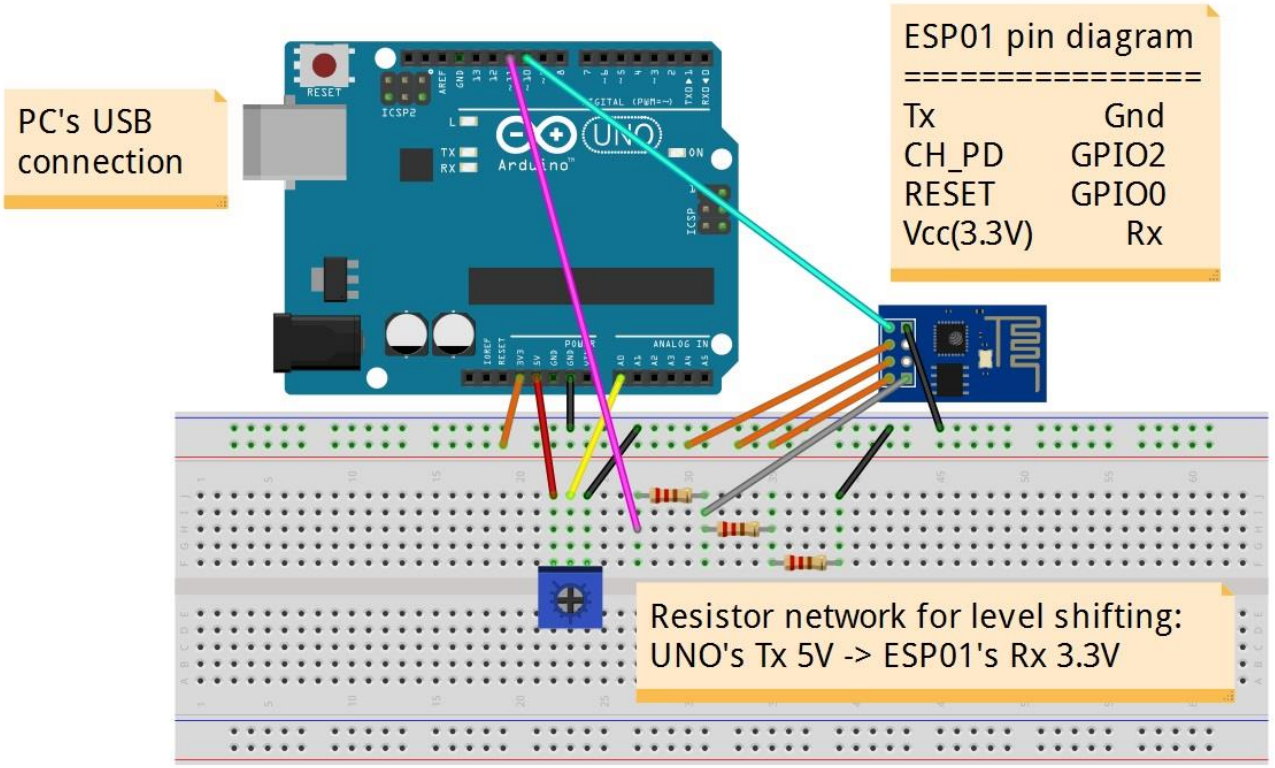
FYI only, no need to go into details.



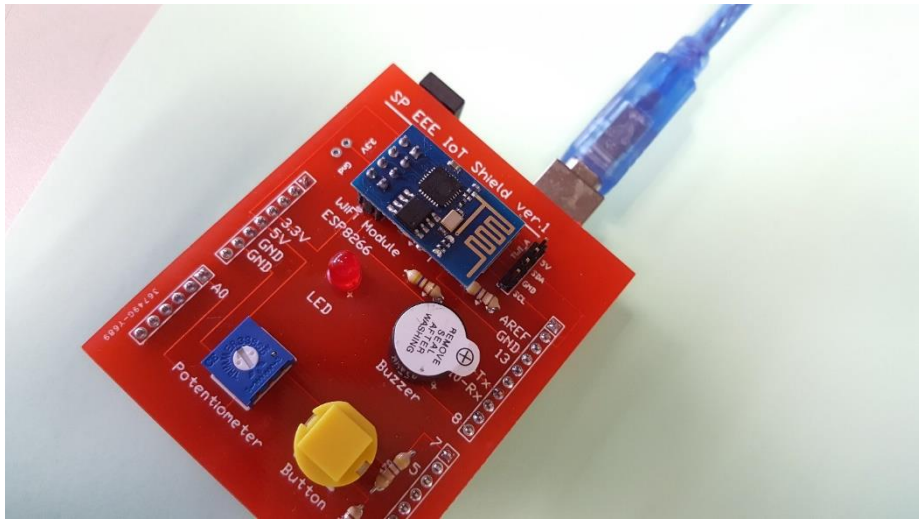
Rx of ESP01 is connected to pin 11 i.e. Tx of UNO. Level shifting is achieved by 3 equal resistors in series, and tapping at the 2/3 point, since 2/3 x 5V = 3.3V.

Exercise 3.2 – Using WiFi module – connections (cont.)

Connection (cont.)



Instead of connecting up using breadboard & wires, we will use a shield instead.



Exercise 3.3 – Using WiFi module – basic AT commands

sketch_test_IO_ESP01 | Arduino 1.8.2

File Edit Sketch Tools Help

Sample Code

sketch_test_IO_ESP01

Download this sample code to the UNO.

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // software serial RX = pin 10 -> connect to Tx of ESP01
        // software serial TX = pin 11 -> connect to resistor network, before going to Rx of ESP01

void setup() {
  pinMode(8, OUTPUT); // Buzzer
  pinMode(5, OUTPUT); // LED
  pinMode(7, INPUT);  // Button
                      // A0 is Potentiometer

  Serial.begin(9600);
  while (!Serial) {
  }
  Serial.println("Please press button, turn potentiometer or enter AT commands...");
  mySerial.begin(9600);
}
```

Sample code continues on the next page.

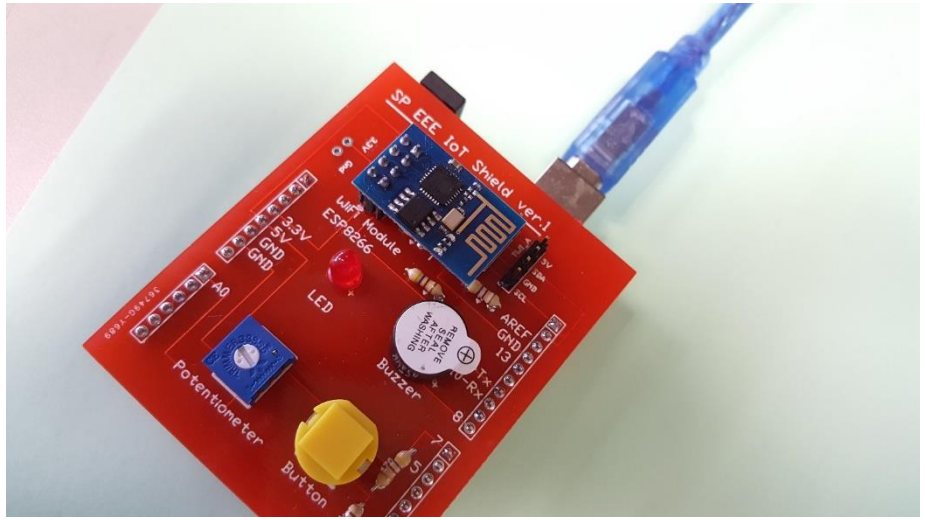
Exercise 3.3 – Using WiFi module – basic AT commands (cont.)

Sample Code (cont.)

```
void loop() {
  // This part allows you to test the button, buzzer, potentiometer & LED:
  if (digitalRead(7)==HIGH)
    digitalWrite(8, HIGH);
  else
    digitalWrite(8, LOW);
  int potentiometerValue = analogRead(A0); // 10-bit: 0-1023
  int LEDbrightness = potentiometerValue / 4; // 8-bit: 0-255
  analogWrite(5, LEDbrightness);

  // This part allows you to issue AT commands from serial monitor, to test ESP01:
  if (Serial.available()) {
    // UNO receives AT commands from hardware serial/PC, and sends to software serial/ESP01.
    mySerial.write(Serial.read());
  }
  if (mySerial.available()) {
    // UNO receives responses from software serial/ESP01, and sends to hardware serial/PC.
    Serial.write(mySerial.read());
  }
}
```

Exercise 3.3 – Using WiFi module – basic AT commands (cont.)



Press the button, and the buzzer will be turned on.

Turn the potentiometer (trimmer), and the LED brightness will change.

Exercise 3.3 – Using WiFi module – basic AT commands (cont.)

1. AT
   expected response: OK
2. AT+RST
   expected response: OK
   @#%... - some "rubbish"
   ready
3. AT+GMR
   expected response: 0018000902-AI03 - firmware version number
   OK
4. AT+CWMODE=1
   expected response: no change - ESP01 is a "station", MODE 2 is AP, MODE 3 is both
5. AT+CWLAP
   expected response: - many access points / networks in the vicinity will be listed
   OK
6. *Get the Access Point SSID & password from the lecturer now.*
   AT+CWJAP="SSID","password"
   expected response: OK
7. AT+CIFSR
   expected response: 192.168.?.?
   OK - the station has been issued an IP address

Open the Arduino IDE serial monitor, and select "Both CR & NL" and "9600 baud".

Issue the following AT commands, and see if you get the correct responses.

You can obtain the full list of AT commands for ESP01 by searching the internet.

Once the ESP01 has been issued an IP address, it can access the internet.

The AT commands can be issued by writing code, instead of manually entering at the serial monitor.

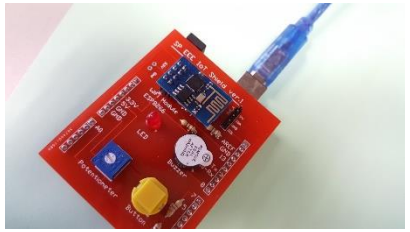
Exercise 3.4 – Using WiFi module – uploading sensor data to the cloud

Applications

- Remote monitoring

Intro

- In this exercise, the sample UNO code used will issue **AT commands** to the ESP01, to **configure** it. After that, the potentiometer (i.e. our **sensor**) will be read at regular **intervals**, and the readings uploaded to **Thingspeak** cloud platform.
- The lecturer will demonstrate how a Thingspeak **account & channel** can be created.
- Some **customizations** to the sample code will be needed – apiKey, SSID & password.



Once the program is running, turn the potentiometer and see how the uploaded sensor data changes.

Exercise 3.4 – …uploading sensor data to the cloud (cont.)

Sample Code

Download this sample code to the UNO.

```
#include <SoftwareSerial.h>

#define DEBUG true

// LED at D13
int ledPin = 13;

// Potentiometer at A0
int potPin = 0;

// replace with your Thingspeak channel's API key!!!
String apiKey = "ABCD1234EFGH"; // *** Change this!

// UNO's D10 connected to ESP's TX
// UNO's D11 connected to ESP's RX via resistor network
SoftwareSerial ESP01(10, 11); // RX, TX
```

```
void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
  while (!Serial){
  }
  Serial.println("Starting...");
  ESP01.begin(9600);
}
```

Customize!

Sample code continues on the next page.

Exercise 3.4 – …uploading sensor data to the cloud (cont.)

Sample Code (cont.)

Customize!

```
void loop() {
  // Reset ESP8266, put it into mode 1 i.e. STA only, make it join hotspot / AP,
  // establish single connection
  Serial.println();
  sendData("AT+RST\r\n",2000,DEBUG);
  sendData("AT+CWMODE=1\r\n",2000,DEBUG);
  sendData("AT+CWJAP=\"EEE-IoT\",\"howIknow@07\"\r\n",4000,DEBUG);
   // *** Change these!
  sendData("AT+CIPMUX=0\r\n",2000,DEBUG);

  // Blink LED on board
  digitalWrite(ledPin, HIGH);
  delay(200);
  digitalWrite(ledPin, LOW);

  // Read potentiometer value
  int sensorValue = analogRead(A0); // 10 bit result: 0 - 1023
  float voltage = sensorValue * (5.0 / 1023.0); // 0V - 5V
  String temp = String(voltage); // convert to string
  Serial.println(temp);
```

Sample code continues on the next page.

Exercise 3.4 – …uploading sensor data to the cloud (cont.)

Sample Code (cont.)

```
// Make TCP connection
String cmd = "AT+CIPSTART=\"TCP\",\"";
cmd += "184.106.153.149"; // Thingspeak.com's IP address
cmd += "\",80\r\n";
sendData(cmd,2000,DEBUG);

// Prepare GET string
String getStr = "GET /update?api_key=";
getStr += apiKey;
getStr +="&field1=";
getStr += temp;
getStr += "\r\n";
```

```
// Send data length & GET string
ESP01.print("AT+CIPSEND=");
ESP01.println (getStr.length());
Serial.print("AT+CIPSEND=");
Serial.println (getStr.length());
delay(500);
if( ESP01.find( ">" ) )
{
  Serial.print(">");
  sendData(getStr,2000,DEBUG);
}

// Close connection, wait a while before repeating...
sendData("AT+CIPCLOSE",16000,DEBUG); // thingspeak needs 15 sec delay between updates
}
```

Refer next page to see how TCP uses Connect + Transfer Data + Disconnect.

TCP: Connect + transfer data + disconnect

Connect: AT + CIPSTART = "TCP", “184.106.153.149“, 80

Transfer data: GET /update?api_key=ABCD1234EFGH&field1=1.23

AT + CIPSEND = 56 // length of GET string to be sent
// Wait for the > prompt… then send the GET string
> GET /update?api_key=ABCD1234EFGH&field1=1.23

Disconnect: AT + CIPCLOSE

Sample code continues on the next page.

Exercise 3.4 – …uploading sensor data to the cloud (cont.)

Sample Code (cont.)

```
String sendData(String command, const int timeout, boolean debug)
{
    String response = "";
    ESP01.print(command);
    long int time = millis();
    while( (time+timeout) > millis())
    {
      while(ESP01.available())
      {
        // "Construct" response from ESP01 as follows
         // - this is to be displayed on Serial Monitor.
        char c = ESP01.read(); // read the next character.
        response+=c;
      }
    }
    if(debug){
      Serial.print(response);
    }
    return (response);
}
```

A function to send out the AT commands and to read the ESP01's response.


Sending a very big
THANK YOU!