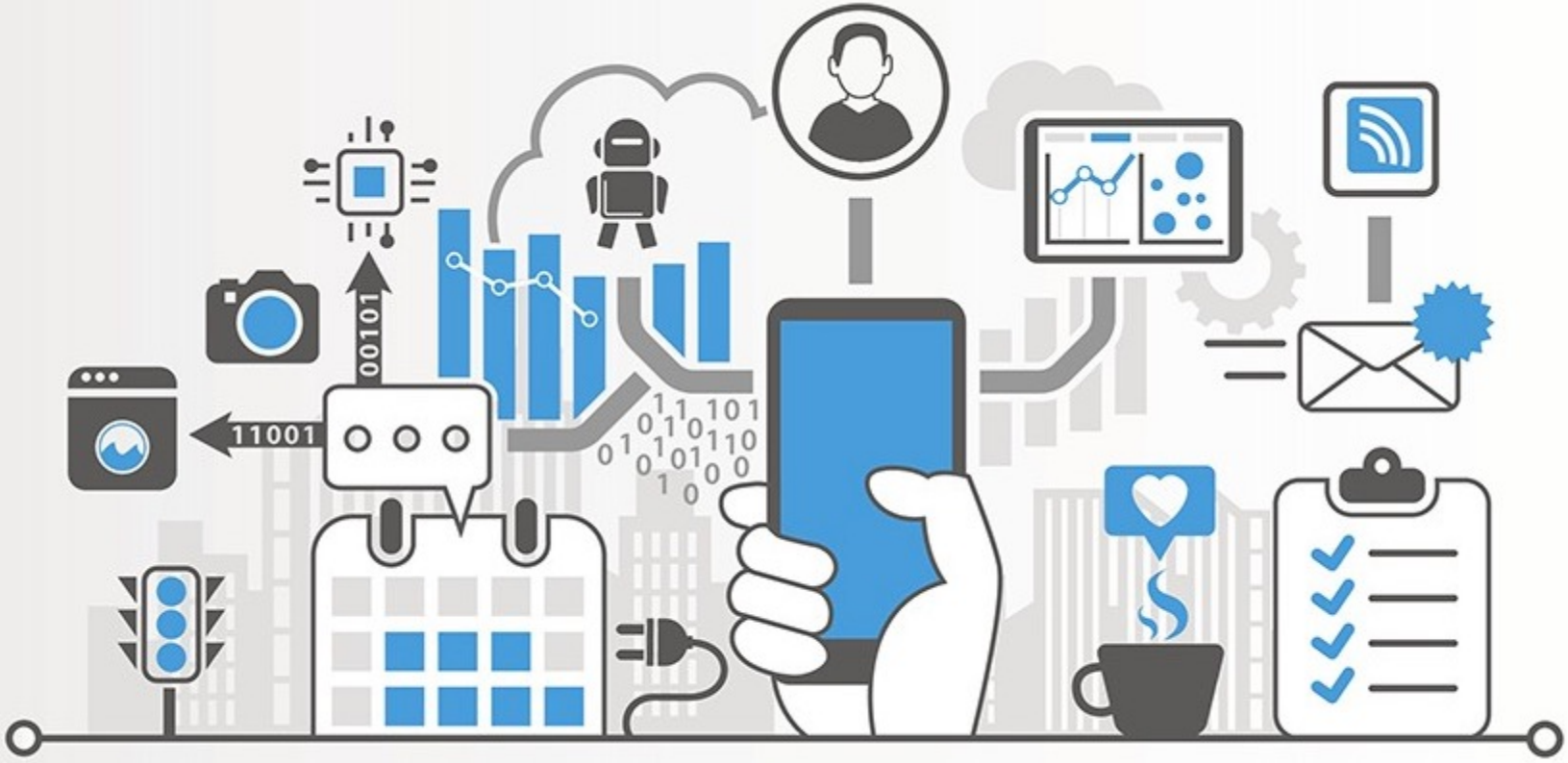
Tecnologie per IoT

Daniele Jahier Pagliari

Lab1: Hardware



PART2: EXERCISE 1

Exercise 1

- Read specification on the lab PDF…
  - Some further details here

Exercise 1

- Spec. 1)
  - To reduce the number of wires and connections, you can also use the internal temperature sensor of the Arduino…
  - Although less precise and accurate than the external one.

Parenthesis: LSM6DSOXTR

- The board includes a LSM6DSOXTR Inertial Measurement Unit (IMU) from STM.
  - 3D Accelerometer + 3D Gyroscope + Temperature Sensor

- We can use the IMU functionality by including the corresponding library:

```
1    #include <Arduino_LSM6DSOX.h>
```

Parenthesis: LSM6DSOXTR

- In the setup() we must initialize the library as follows:

```
7   if (!IMU.begin()) {
8     Serial.println("Failed to initialize IMU!");
9     while(1);
10  }
```

- Lastly, we can read the temperature (directly in Celsius, as an int) with the following code:
  - Drawback: the resolution is 1 degree, and the sensor less accurate than the external one.

```
14  if (IMU.temperatureAvailable()) {
15    int temperature = 0;
16    IMU.readTemperature(temperature);
17  }
```

Exercise 1

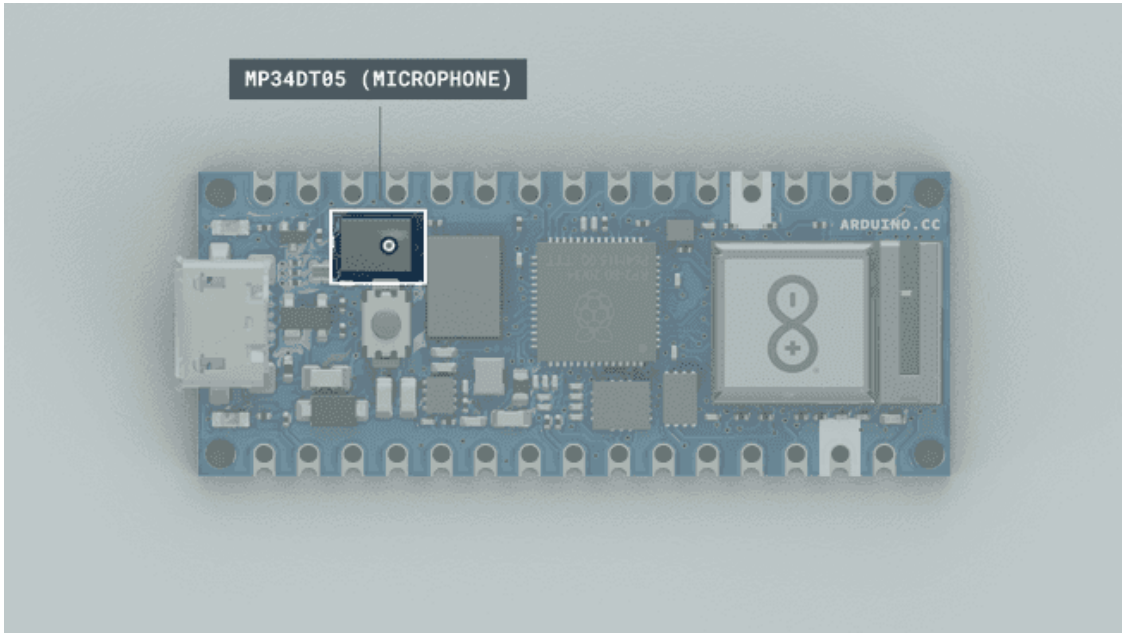
- Spec. 2)
  - LED light is proportional to current
  - With the circuit of Ex. 1.1, we can regulate the current by reducing the voltage drop
  - **We can control LED intensity using PWM!**

Exercise 1

- Spec. 4)
  - We didn't look at the microphone yet…

Microphone

- The RP2040 Connect has an on-board MP35DT05 microphone that uses PDM (Pulse-Density Modulation)
  - PDM is a more general case of PWM with variable period.
  - Datasheet.



Microphone

- Luckily the PDM arduino library does the PDM-to-digital conversion for you
- You should use this library to interact with the microphone
- Include the header file:

```
1    #include <PDM.h>
```

- In the globals, define a buffer to store the audio samples (converted to 16bit digital values). By default, the library expects a 512 bytes buffer. This can be changed with (setBufferSize()):

```
14   // Buffer to read samples into, each sample is 16-bits
15   volatile short sampleBuffer[256];
```

Microphone

- In the setup(), associate a callback to execute everytime new data arrives

Callback function (ISR)

```
22    PDM.onReceive(onPDMdata);
```

- Then initialize the PDM sensor:

N. Channels (1 = mono) Sample frequency

```
23    if (!PDM.begin(1, 16000)) {
24      Serial.println("Failed to start PDM!");
25      while (1);
26    }
```

Microphone

- ISR function:
  - Check how many bytes are available
  - Read them in the buffer
  - Remember that each sample requires 2 bytes
  - Note: it's an ISR, so you can't call delay(), Serial.print(), etc.

```
64  void onPDMdata() {
65    // Query the number of available bytes
66    int bytesAvailable = PDM.available();
67    // Read into the sample buffer
68    PDM.read(sampleBuffer, bytesAvailable);
69    // 16-bit, 2 bytes per sample
70    samplesRead = bytesAvailable / 2;
71  }
```

Microphone

- In the loop(), use the buffer values as needed…

```
63  void loop() {
64    if (samplesRead) {
65      for (int i = 0; i < samplesRead; i++) {
66        Serial.println(sampleBuffer[i]);
67      }
68      samplesRead = 0;
69    }
70  }
```

- Note: this is an example, implementing the lab request is more tricky!!

Exercise 1

- Spec. 6)
  - When presence is detected:
    - $T_{AC,min} = T_{AC,min,pres}$
    - $T_{AC,max} = T_{AC,max,pres}$
    - etc. (same for heater)
  - Otherwise:
    - $T_{AC,min} = T_{AC,min,abs}$
    - $T_{AC,max} = T_{AC,max,abs}$
    - etc. (same for heater)

Exercise 1

- The rest is up to you…