Introduction to Embedded Systems (IES)

Module 5
Practical use cases

Version 2023-02-13, Kjeld Jensen kjen@sdu.dk

In this module we will use the learning about embedded systems from the previous modules to solve real-world practical use cases. We will work with cases concerning water tower pump control, automatic street lights and a refrigerated medicine temperature alarm.

Agenda

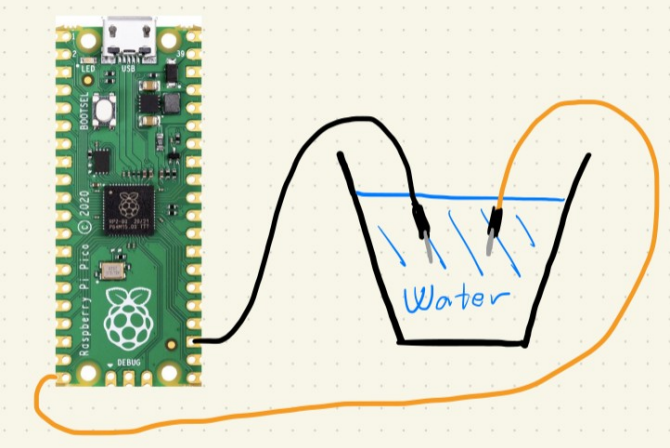
1. Notes
   a) Final report assignment and Questionary will be sent to on WatchApp later this week
2. Review of module 4
3. Use cases
   1) Water tower pump control
   2) Automatic street lights
   3) Refrigerated medicine temperature alarm

A) Water tower pump control

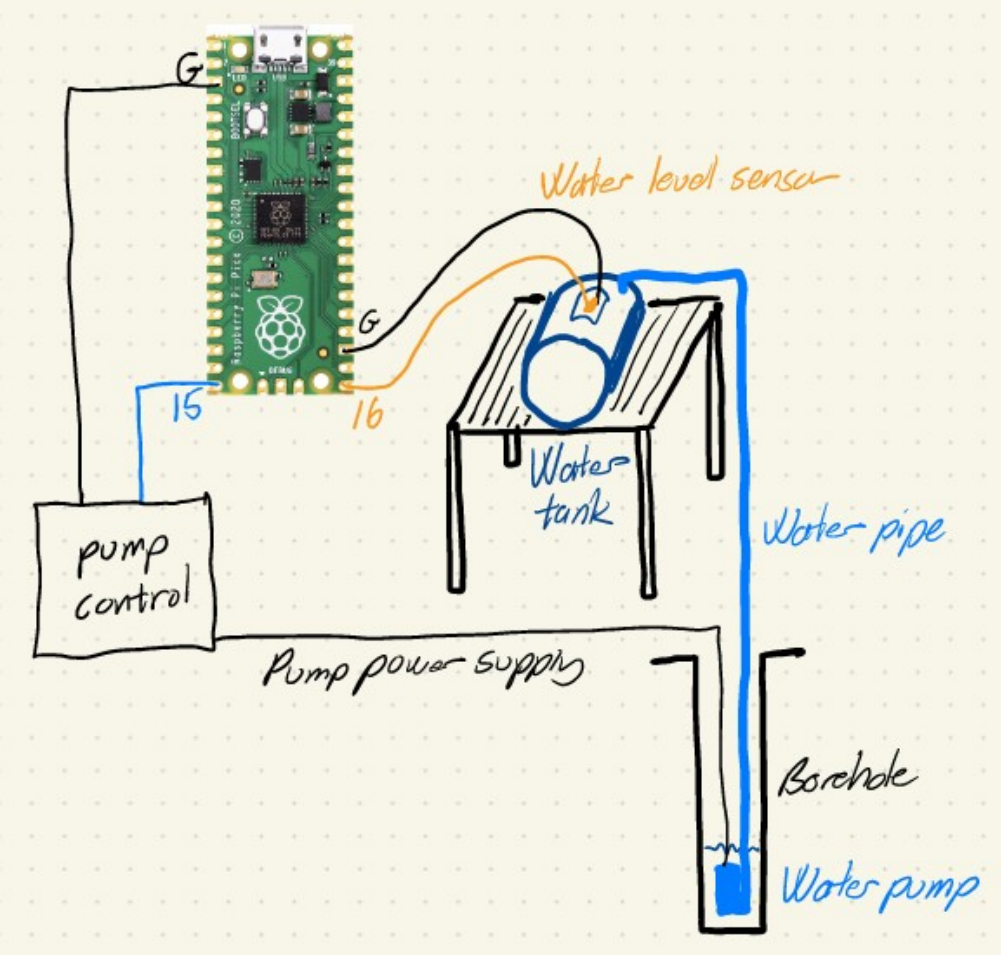
The image below (from Gandorhun) shows a typical water tower providing water to a community.

In many cases the pump caretaker daily turns on the pump when needed, and turns it back off, when the water is seen running out from the top of the tank. This can also be automatized using an embedded system.

In module 3 we created a simple water sensor that gave a digital low signal when there is water available and a digital high signal when there is no water.

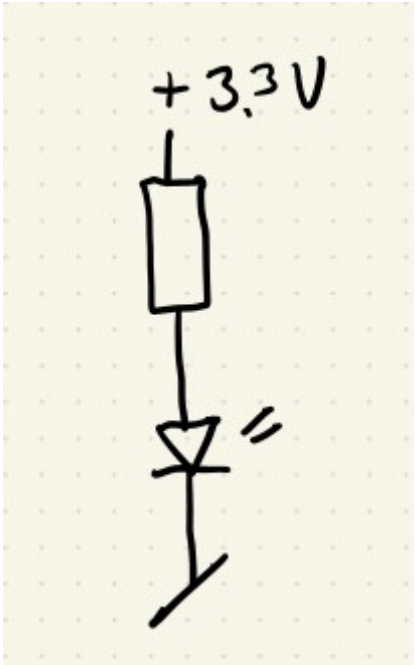


Imagine that we put this sensor at the top of the water tower tank i.e. place the two wires just below the tank top opening. This would allow us to measure if the tank is full or not. If we can control the pump by a digital output, we would then be able to automatize the pumping process like illustrated in the sketch below. The solar cell panels supplying power to the pump control and the pump have been removed for simplicity.

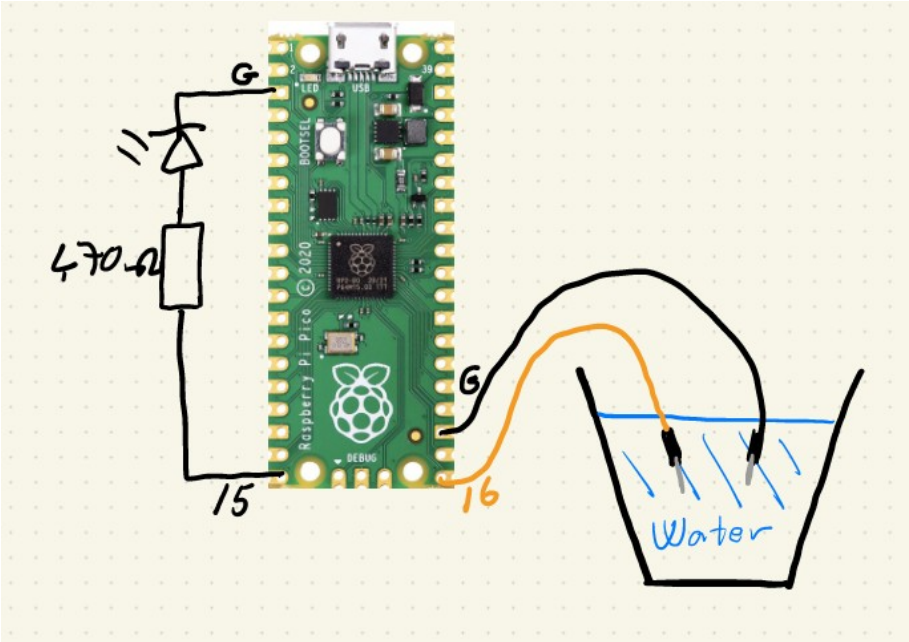


In our course kit we obviously don’t have a water pump or a pump controller. For simulating the water pump we will thus use an external LED. We will use

the same circuit as in exercise 4.A and the same arduino example program `digital_output.ino` to control it.



Exercise A.1) Please connect the water sensor wire to pin 16 on the Pico and the LED to pin 15 on the Pico using this circuit:



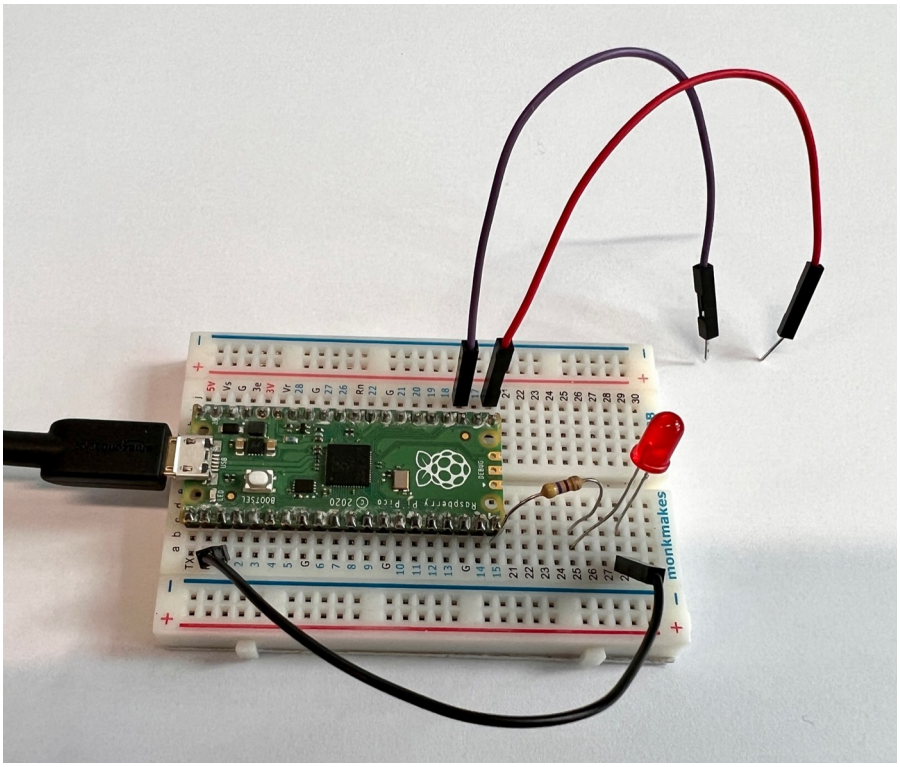
Hint: You don’t need a glass of water for for this module, just short the two wires to simulate that water has been sensed.

Please remember that the **shortest** pin of the LED is the **negative** pin which must be connected to the Ground **G** pin. Please also remember to add a 470 Ohm resistor in series with the LED to limit the current.

The 470 Ohm resistor has the color code:

`yellow - violet - brown - gold`

Here is an example of how the breadboard will look:



Exercise A.2) Please test that the LED works by testing it with the `digital_output.ino` program available in the examples folder.

Exercise A.3) Please test you are able to read the digital value of the water sensor by testing it with the `read_digital_input.ino` program available in the examples folder. Remember to change the input pin number.

```
#define PIN_DIGITAL_IN 15

void setup() {
  // Setup Pins
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(PIN_DIGITAL_IN, INPUT_PULLUP);
}

void loop() {
  // Read pin high/low state
  bool logic_high =
digitalRead(PIN_DIGITAL_IN);

  if (logic_high == true) {
    // Turn off LED
    digitalWrite(LED_BUILTIN, LOW);
  } else {
    // Turn on LED
    digitalWrite(LED_BUILTIN, HIGH);
  }
}
```

Exercise A.4) Based on the two example programs from exercises A.2 and A.3 please write a program that automatically turns off the pump (the external LED) when the tank is full (water is detected) and turns on the pump (LED) when the tank needs to be filled (no water is detected).

Hint: You can use the `if` statement to control the pump (LED) by following this example:

```
if (digitalRead(PIN_DIGITAL_IN) == HIGH) {
  // turn off the pump
} else {
  // turn on the pump
}
```

When done please post a video of your working embedded system on WhatsApp.

B) Automatic street lights

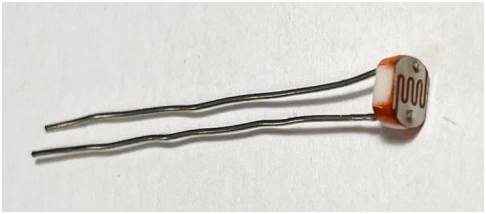
Streets with no street lights at night are often a problem in terms of both safety and security. Putting up street lights is expensive, however, due to the power consumption.

In this exercise we will create an ambedded system that automatically turns on a street light when it is dark.

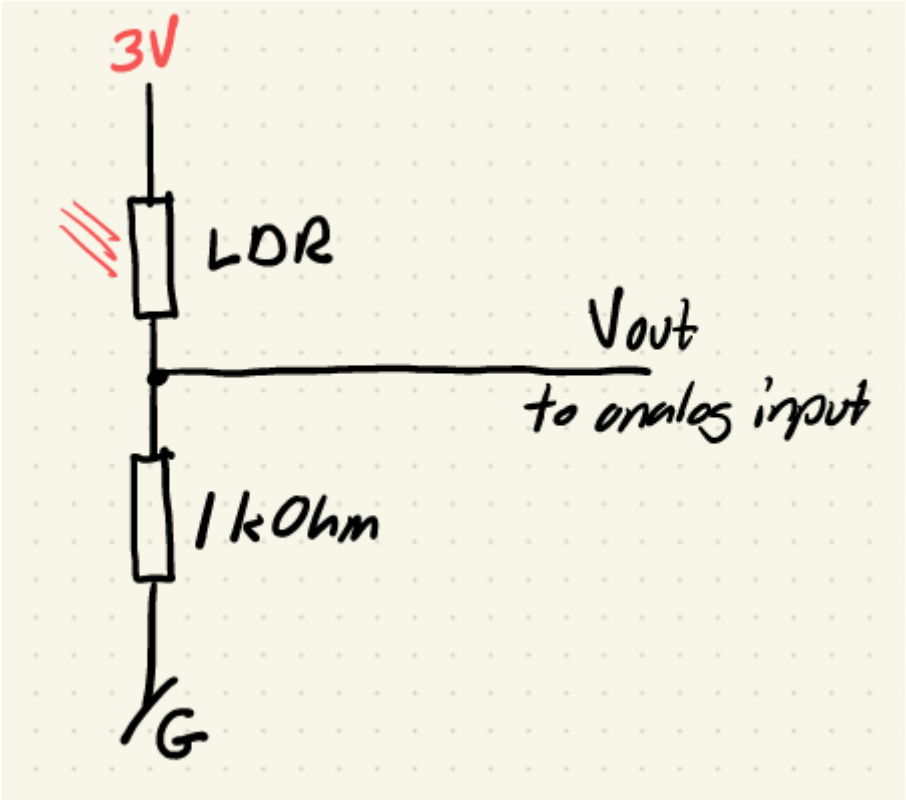
Building upon this exercise it would be possible to turn on the street light only when desired in order to save power. This could for instance be a few hours after sunset and a few hours before sunrise.

In our course kit we obviously don’t have a street light, so like in exercise A we will thus use the external LED to simulate the street light.

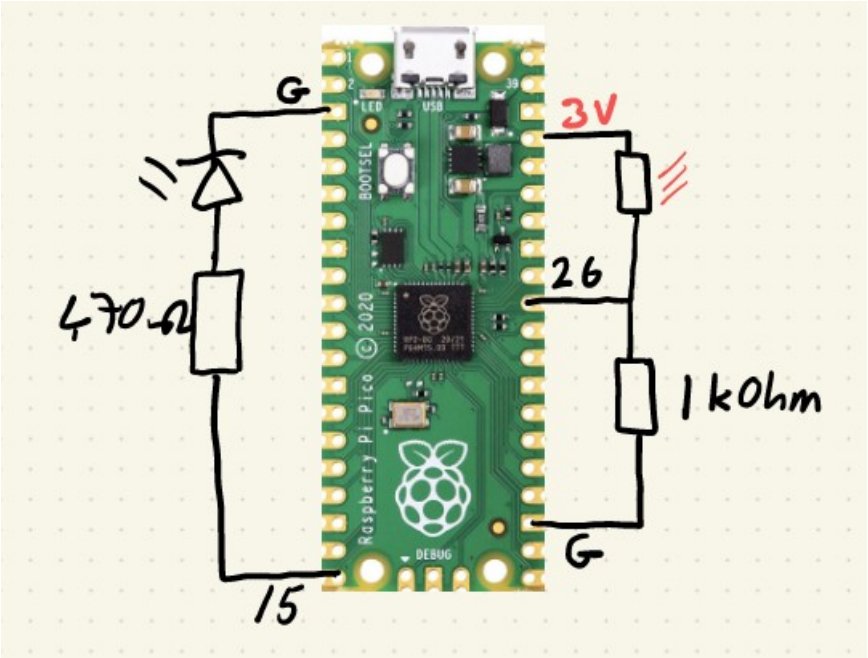
For sensing the light we will use the photoresistor (LDR) which is a light sensitive resistor whose resistance decreases as the intensity of light it is exposed to increases.



The photoresistor acts like the potentiometer in module 3 exercise D and we will use the same voltage divider circuit and the same arduino example program `read_analog_input.ino` to read it.



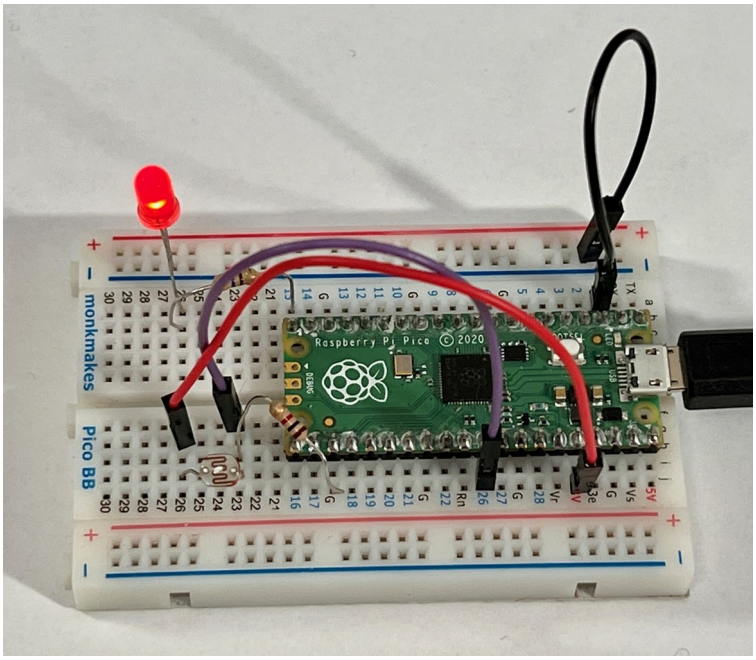
Exercise B.1) Please connect the phototransistor to pin 26 on the Pico and the LED to pin 15 on the Pico using this circuit:



The 1 kOhm resistor has the color code:

`brown - black - red - gold`

Here is an example of how the breadboard will look:



Exercise B.2) Please test that the LED works by testing it with the `digital_output.ino` program available in the examples folder.

```
#define PIN_DIGITAL_OUT 15

void setup() {

  // setup the PIN_DIGITAL_OUT as digital
output
  pinMode(PIN_DIGITAL_OUT, OUTPUT);
}

void loop() {

  // toggle the PIN_DIGITAL_OUT once
  digitalWrite(PIN_DIGITAL_OUT, LOW);
  delay(500);
  digitalWrite(PIN_DIGITAL_OUT, HIGH);

  // wait 2 seconds
  delay(2000);
}
```

Exercise B.3) Please test you are able to read the analog value of the photoresistor by testing it with the `read_analog_input.ino` program available in the examples folder.

```
#define PIN_ANALOG_IN 26

void setup() {
  // Setup serial port
  Serial.begin(115200);
}

void loop() {
  // Read analog input pin
  int value = analogRead(PIN_ANALOG_IN);

  // Print the value to the serial port
  Serial.println(value);

  // Wait one second
  delay(1000);
}
```

Please remember from the exercise in module 3 that when you have uploaded the example program to the Pico, you have to open the Serial Montior (magnifying glass icon to the upper right of the Arduino Programming Software).

You should be able to observe that the value decreases when you put your finger in front of the phototransistor and thus shades the light. Conversely

the value should increase when you expose the phototransistor to light.

Exercise B.4) Based on those two example programs please write a program that automatically turns on the LED when you put your finger in front of the phototransistor, and turns it off when you remove it.

Hint: You can use the if statement to control the LED by following this example. Please notice that you will probably have to adjust the value 250 according to your readings:

```
if (value > 250) {
  // turn off the light
} else {
  // turn on the light
}
```

When done please post a video of your working embedded system on WhatsApp.

C) Refrigerated medicine temperature alarm

A range of medicines such as insuline, antibiotic etc. need to be refrigerated. If the refrigerator[1] looses power or breaks, the medicine will quickly become useless.

In this exercise we will create an alarm system that sounds if the temperature of the refrigerator rises above an acceptable temperature.

Exercise C.1) Please test that you can read the temperature of the RP2040 processor on the Pico board like we did in module 1 using the `read_temperature.ino` program available in the examples folder.

1 Image source: https://www.cleanpng.com/png-vaccine-refrigerator-cold-chain-pharmaceutical-dru-659219/

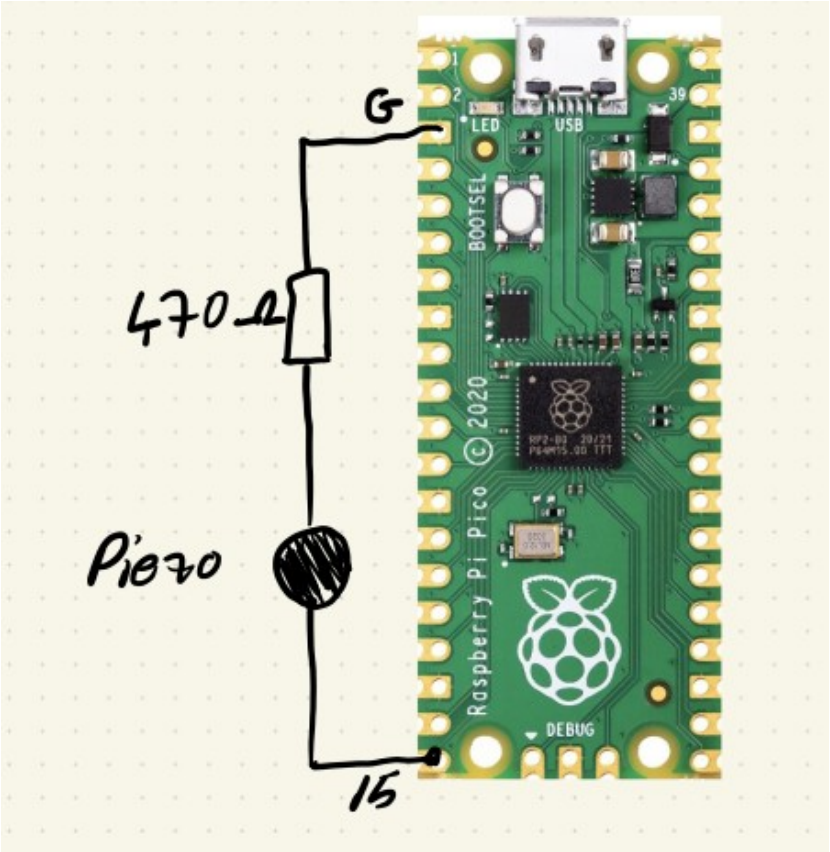
```
void setup() {
  Serial.begin(115200);
  delay(2000);
}

void loop() {
  double temperature;

  // read the temperature in degrees Celcius
  temperature = analogReadTemp();

  // print the temperature to the serial port
  Serial.printf("Pico core temperature is:
%2.1fC\n", temperature);

  // wait one second
  delay(1000);
}
```

Please remember that when you have uploaded the example program to the Pico, you have to open the Serial Montior (magnifying glass icon to the upper right of the Arduino Programming Software).

When you see the temperature, gently press your finger against the RP2040 chip on the Pico board. You should see that the temperature increases because your finger is warming it up.

When you are able to read the temperature, it is time to create the alarm circuit.

Exercise C.2) Please connect the piezo horn to pin 15 on the Pico like in module 4 exercise D. Please remember to add a 470 Ohm resistor in series with the piezo buzzer to limit the current.



The 470 Ohm resistor has the color code:

`yellow - violet - brown - gold`

Exercise C.3) Please test that you can use the example program from module 4 exercise D to make a piezo buzzer sound. A copy is available as `piezo_control.ino` in the examples folder.

```
#define PIN_PIEZO 15

void beep(unsigned char ms){

  // turn on the piezo buzzer, almost any value
between 1 and 254 can be used
  analogWrite(PIN_PIEZO, 100);

  // wait ms milliseconds
  delay(ms);

  // turn off the piezo buzzer
  analogWrite (PIN_PIEZO, 0);

  // wait ms milliseconds
  delay(ms);
}

void setup() {

  // setup the PIN_PIEZO as output
  pinMode(PIN_PIEZO, OUTPUT);
}

void loop() {
  beep(200);
  beep(100);
}
```

Exercise C.4) Based on those two example programs please write a program that automatically sounds an alarm when the temperature rises above a defined threshold.

For a refrigerator the threshold would probably be about 8 degrees Celcius, however for this exercise you can select a threshold that is two degrees above the room temperature where you are. Then you can easily reach the threshold by pressing your finger gently against the RP2040 processor.

Hint: You can use the if statement to control the LED by following this example:

```
#define PIN_DIGITAL_OUT 15

void setup() {

  // setup the PIN_DIGITAL_OUT as digital
output
  pinMode(PIN_DIGITAL_OUT, OUTPUT);
}

void loop() {

  // toggle the PIN_DIGITAL_OUT once
  digitalWrite(PIN_DIGITAL_OUT, LOW);
  delay(500);
  digitalWrite(PIN_DIGITAL_OUT, HIGH);

  // wait 2 seconds
  delay(2000);
}
```