# Exercise 4: The Micro:bit LED Screen

## Lab IoT

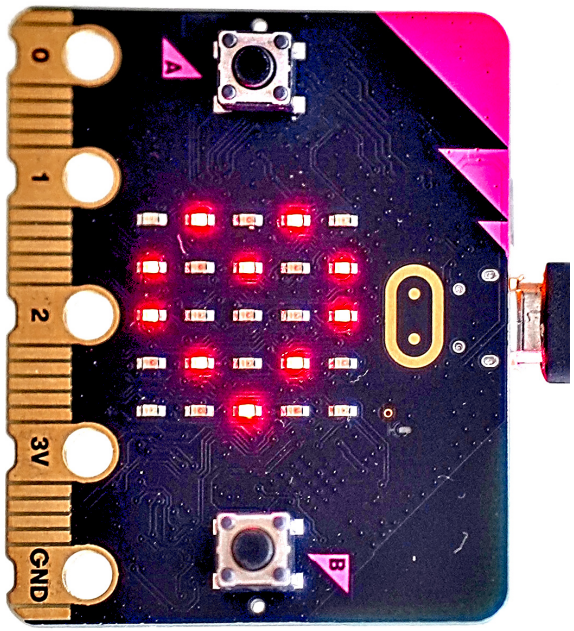
Philipp H. Kindt

Assistant Professorship for Pervasive Computing Systems (PCS)
TU Chemnitz

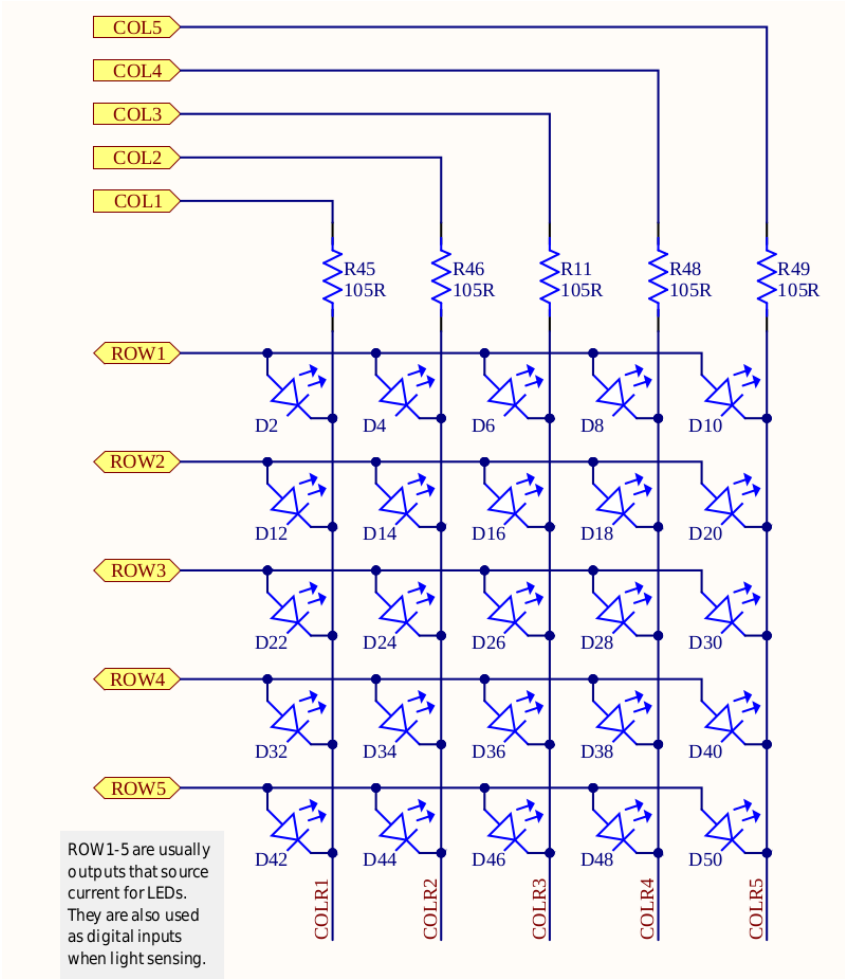
November 11, 2021

# Goals

- The goal of this exercise is to use the LED matrix (screen) on the Micro:bit board.
- It will involve timers, interrupts and the GPIO.
- Some code will be given, but it's not yet complete.
- Your task will be to complete the code, such that a heart is shown on the screen



# The Micro:bit Screen




Source: Micro:bit Educational Foundation. Micro:bit v2 schematic. CC-BY-SA-4.0 License. `https://raw.githubusercontent.com/microbit-foundation/microbit-v2-hardware/main/V2/MicroBit_V2.0.0_S_schematic.PDF`

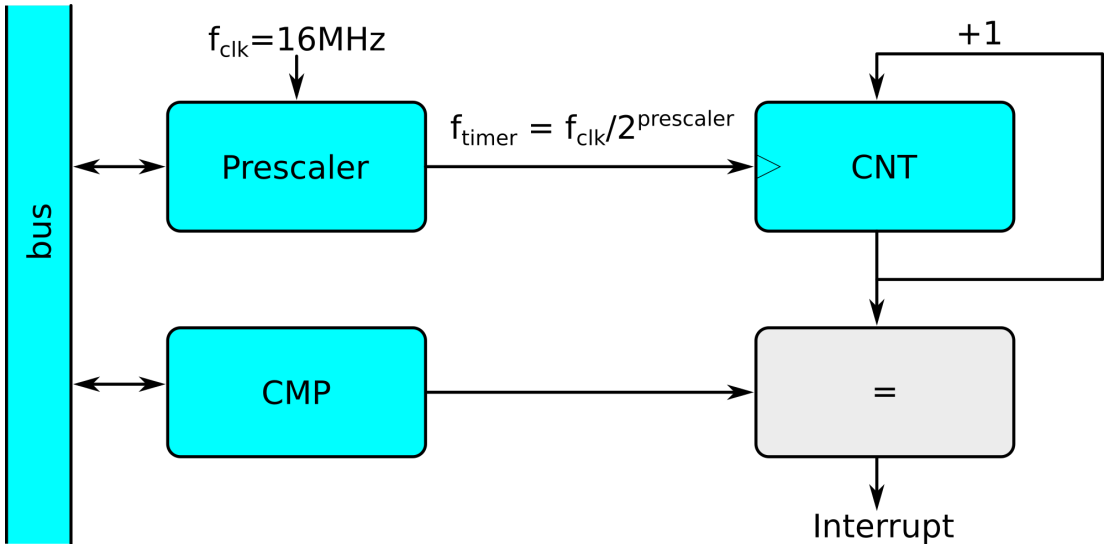
# Driving the LED Matrix

- Each ROW and COL signal is connected to a GPIO pin and can be either driven to 0 (i.e., 0 V) or 1 (i.e., 3 V).
- A LED will shine if the corresponding ROW signal is high and the corresponding COL signal is low.
- Not all combinations of active/inactive LEDs can be realized. E.g., activating D2 and D14 requires (ROW1->1, ROW2->1, COL1->0, COL2->0), whereas activating D4 and D12 requires (ROW1->0, ROW2->0 or COL1->0, COL2->0).
- Is there a way to nevertheless display every possible pattern?

# Driving the LED Matrix - Solution

- Either rows or columns need to be activated *sequentially*, such that only one of them is active at the same time.
- Given an active row (or column), each column (or row) can be activated/deactivated without any constraints.
- The update frequency needs to be sufficiently high, e.g., 500 Hz.
- A notion of time is needed $=>$ The *timer* hardware is used.

# Timers




- A *timer* is a counter, counting ticks of a prescaled clock.
- The *prescaler* is a register used to slow down the frequency $f_{timer}$.
- Once the counter has reached *CMP*, an interrupt is generated.
- The values of *CMP* and *prescaler* can be written via the bus.
- A periodic mode of operation is reached by increasing *CMP* every time the interrupt is generated.

# Using the Timer

- In our case, the timer is controlled through multiple library functions from the *blessed* project that hide the underlying complexity.
- *timer_init()* configures the mode of operation of the timer and sets the prescaler.
- *timer_create()* creates a *handle* to control a particular instance of a timer (or, more precisely, one particular out of multiple *CMP* registers connected to the same counter).
- *timer_start(int16_t id, uint32_t us, timer_cb_t cb)* starts the timer with the handle *id*. The parameter *us* is the period in microseconds, *cb* is a pointer to a function, which is called whenever the timer has expired. Recall that in C, a pointer to a function is equivalent to the function name – without the parentheses "()" at the end.

# Task

- Open the project “lab4_and_lab5”.
- The GPIOs are already initialized. Make yourself familiar with the code.
- A function *update_display()* exists. It switches on the LEDs for the first column, while all other columns remain dark. It is called once per second using a timer, which is already initialized.
- Extend this function to generate the output for all columns, such that a heart is shown on the Micro:bit led display.
- You might need a static variable to “count” the number of calls to this function and select the right row.
- The update frequency of the display is too low. Increase it for making the heart “persistent”.
- After having implemented the code displaying the heart on the Micro:bit screen, extend this function to scroll through the text from left to right. "TUC" or "UNC" on the LED display.