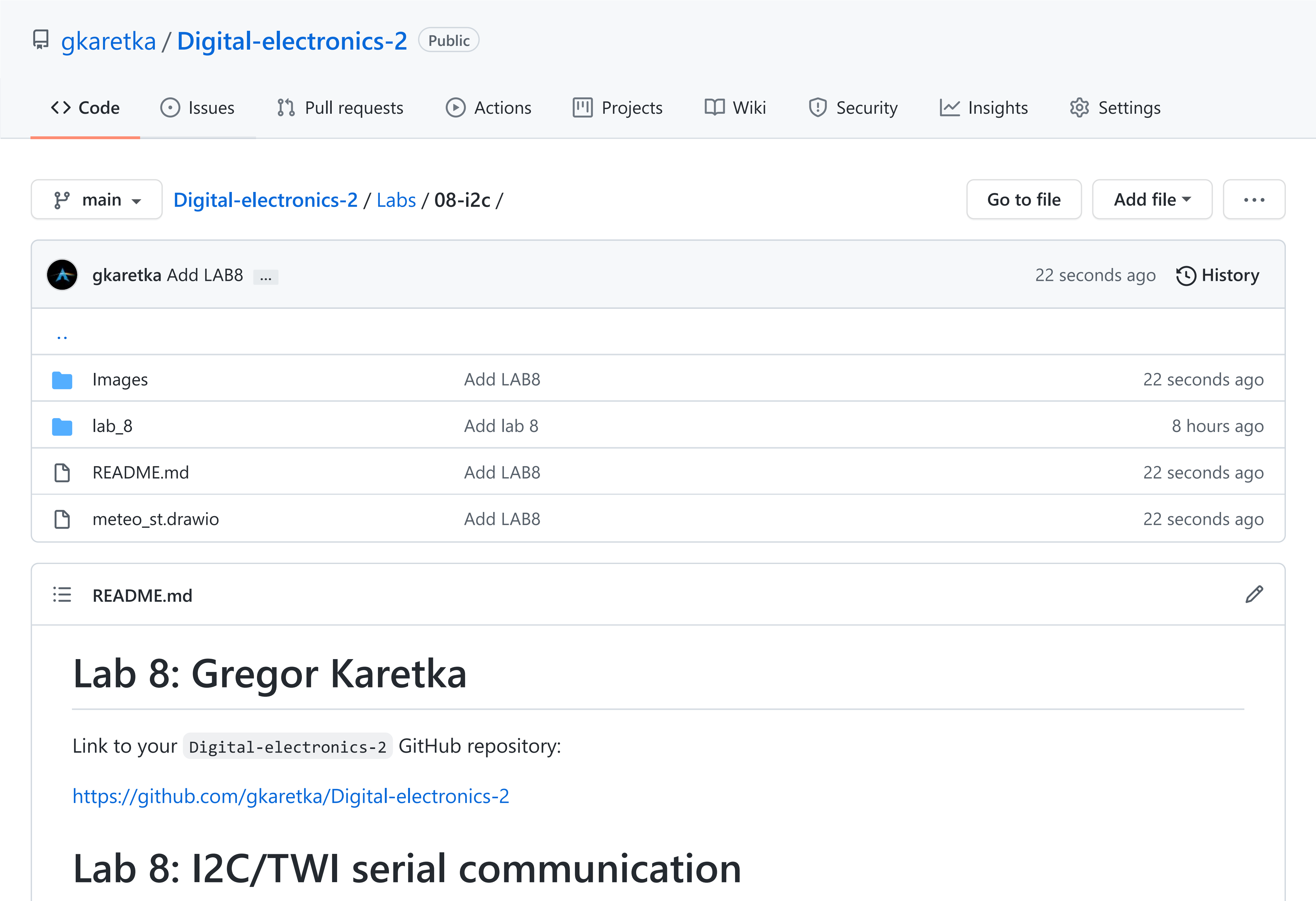


Images	Add LAB8	22 seconds ago
lab_8	Add lab 8	8 hours ago
README.md	Add LAB8	22 seconds ago
meteo_st.drawio	Add LAB8	22 seconds ago

Lab 8: Gregor Karetka

Link to your `Digital-electronics-2` GitHub repository:

https://github.com/gkaretka/Digital-electronics-2

Lab 8: I2C/TWI serial communication

Preparation tasks (done before the lab at home)

1. Use schematic of the Arduino Uno board and find out to which pins the SDA and SCL signals are connected.

Signal	MCU pin	Arduino pin(s)
SDA (data)	PC4	A4/SDA
SCL (clock)	PC5	A5/SCL

2. What is the general structure of I2C address and data frames?

Frame type	8	7	6	5	4	3	2	1	0	Description
Address	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	R/W bit	ACK/NACK	Address of device the communication is started with. ACK by slave with correct address present on the line.
Data	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	ACK/NACK	Data to be send. ACK by slave if data received otherwise NACK.

i. Use the `twi.h` header file from the I2C/TWI library to complete the description of the functions in the following table.

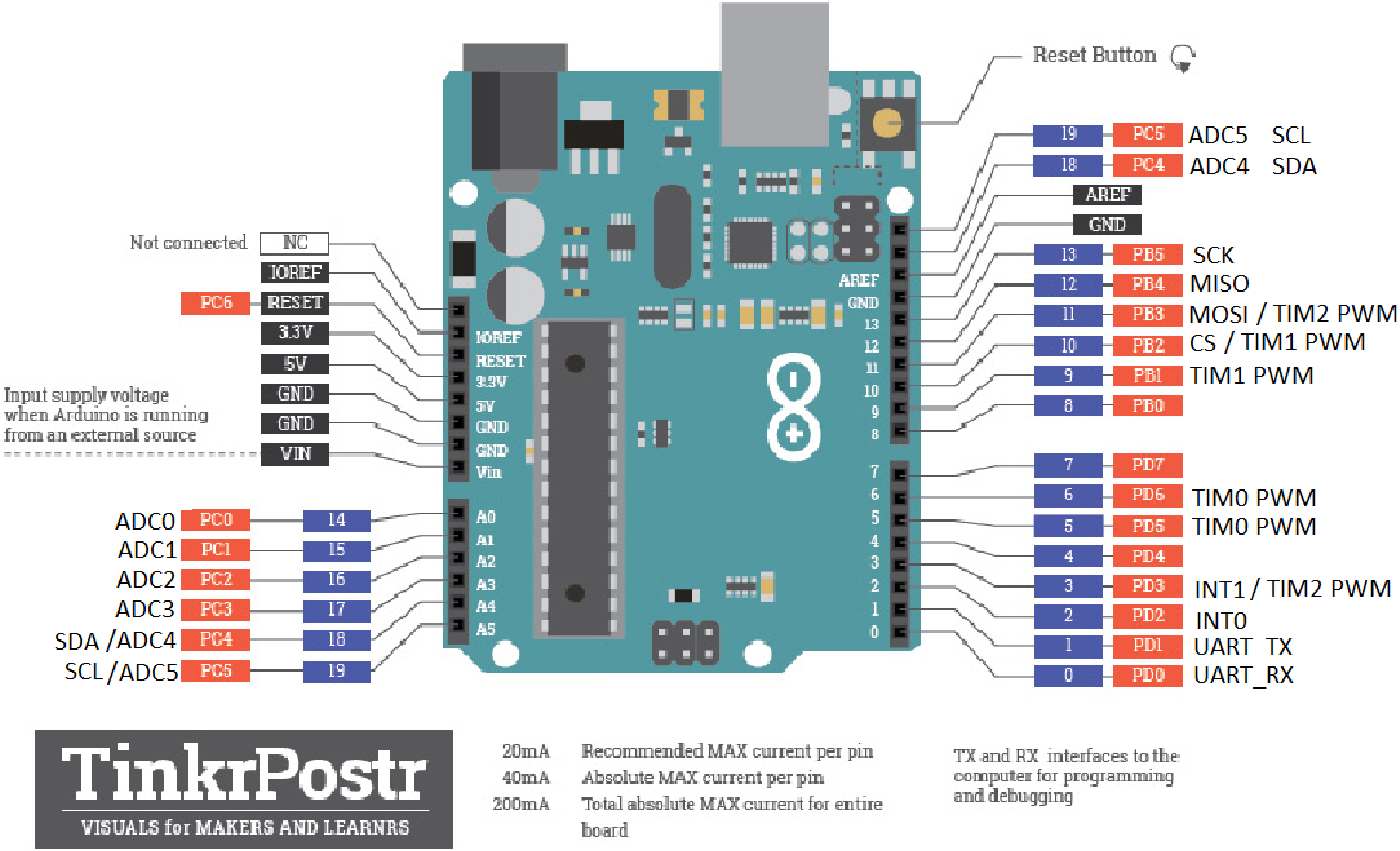
Function name	Function parameters	Description	Example
`twi_init`	None	Initialize TWI, enable internal pull-up resistors, and set SCL frequency	`twi_init();`

Function name	Function parameters	Description	Example
`twi_start`	Device address	Start communication on TWI bus and send address of TWI slave.	`twi_start((addr<<1)+TWI_READ);`
`twi_write`	8 bit data to be sent`	Send one data byte to TWI slave device.	`twi_write(127);`
`twi_read_ack`	None	Read one byte from TWI slave device and acknowledge it by ACK.	`twi_read_ack();`
`twi_read_nack`	None	Read one byte from TWI slave device and acknowledge it by NACK.	`twi_read_nack();`
`twi_stop`	None	Generates stop condition on TWI bus.	`twi_stop();`

Arduino Uno pinout

1. In the picture of the Arduino Uno board, mark the pins that can be used for the following functions:

    - PWM generators from Timer0, Timer1, Timer2
    - analog channels for ADC
    - UART pins
    - I2C pins
    - SPI pins
    - external interrupt pins INT0, INT1

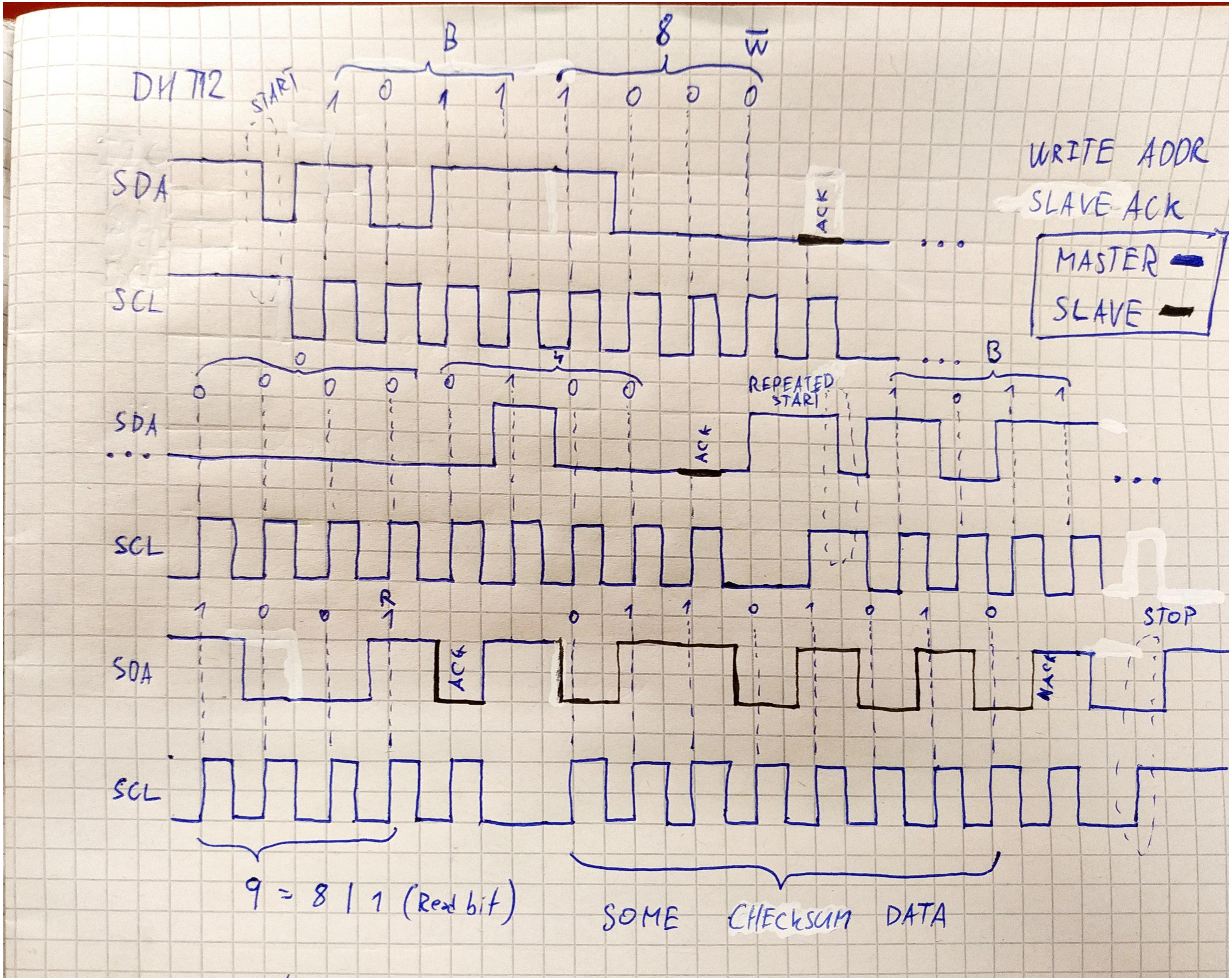


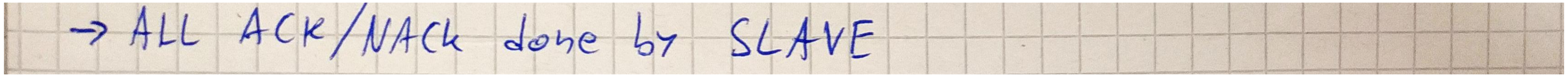
I2C

1. Code listing of Timer1 overflow interrupt service routine for scanning I2C devices and rendering a clear table on the UART.

```
/**********************************************************************
 * Function: Timer/Counter1 overflow interrupt
 * Purpose:  Update Finite State Machine and test I2C slave addresses
 *           between 8 and 119.
 **********************************************************************/
ISR(TIMER1_OVF_vect)
{
    static state_t state = STATE_IDLE;  // Current state of the FSM
    static uint8_t addr = 7;            // I2C slave address
    uint8_t result = 1;                 // ACK result from the bus
    char uart_string[2] = "00";         // String for converting numbers by itoa()

    // FSM
    switch (state)
    {
    // Increment I2C slave address
    case STATE_IDLE:
        // If slave address is between 8 and 119 then move to SEND state
        addr++;
        if (addr >= 8 && addr <= 119) {
            state = STATE_SEND;
        } else if (addr == 0) {
            uart_puts("\n\r\n\r");
        }
        break;

    // Transmit I2C slave address and get result
    case STATE_SEND:
        // I2C address frame:
        // +------------------------+------------+
        // |      from Master       | from Slave |
        // +------------------------+------------+
        // | 7  6  5  4  3  2  1  0 |     ACK    |
        // |a6 a5 a4 a3 a2 a1 a0 R/W|   result   |
        // +------------------------+------------+
        result = twi_start((addr<<1) + TWI_WRITE);
        twi_stop();

        /* Test result from I2C bus. If it is 0 then move to ACK state,
         * otherwise move to IDLE */
        if (result) {
            state = STATE_IDLE;
        } else {
            state = STATE_ACK;
        }

        break;

    // A module connected to the bus was found
    case STATE_ACK:
        // Send info about active I2C slave to UART and move to IDLE
        itoa(addr, uart_string, 10);
        uart_puts(uart_string);
        uart_puts("\n\r");

        state = STATE_IDLE;
        break;

    // If something unexpected happens then move to IDLE
    default:
        state = STATE_IDLE;
        break;
    }
}
```

2. (Hand-drawn) picture of I2C signals when reading checksum (only 1 byte) from DHT12 sensor. Indicate which specific moments control the data line master and which slave.


DHT12
START
B
8
W̄
1 0 1 1 1 0 0 0
SDA
SCL
ACK
...
WRITE ADDR
SLAVE ACK
MASTER
SLAVE
0
4
0 0 0 0 0 1 0 0
SDA
SCL
ACK
REPEATED START
B
1 0 1 1
...
R
1 0 0 1
SDA
SCL
ACK
0 1 1 0 1 0 1 0
NACK
STOP
9 = 8 | 1 (Read bit)
SOME CHECKSUM DATA

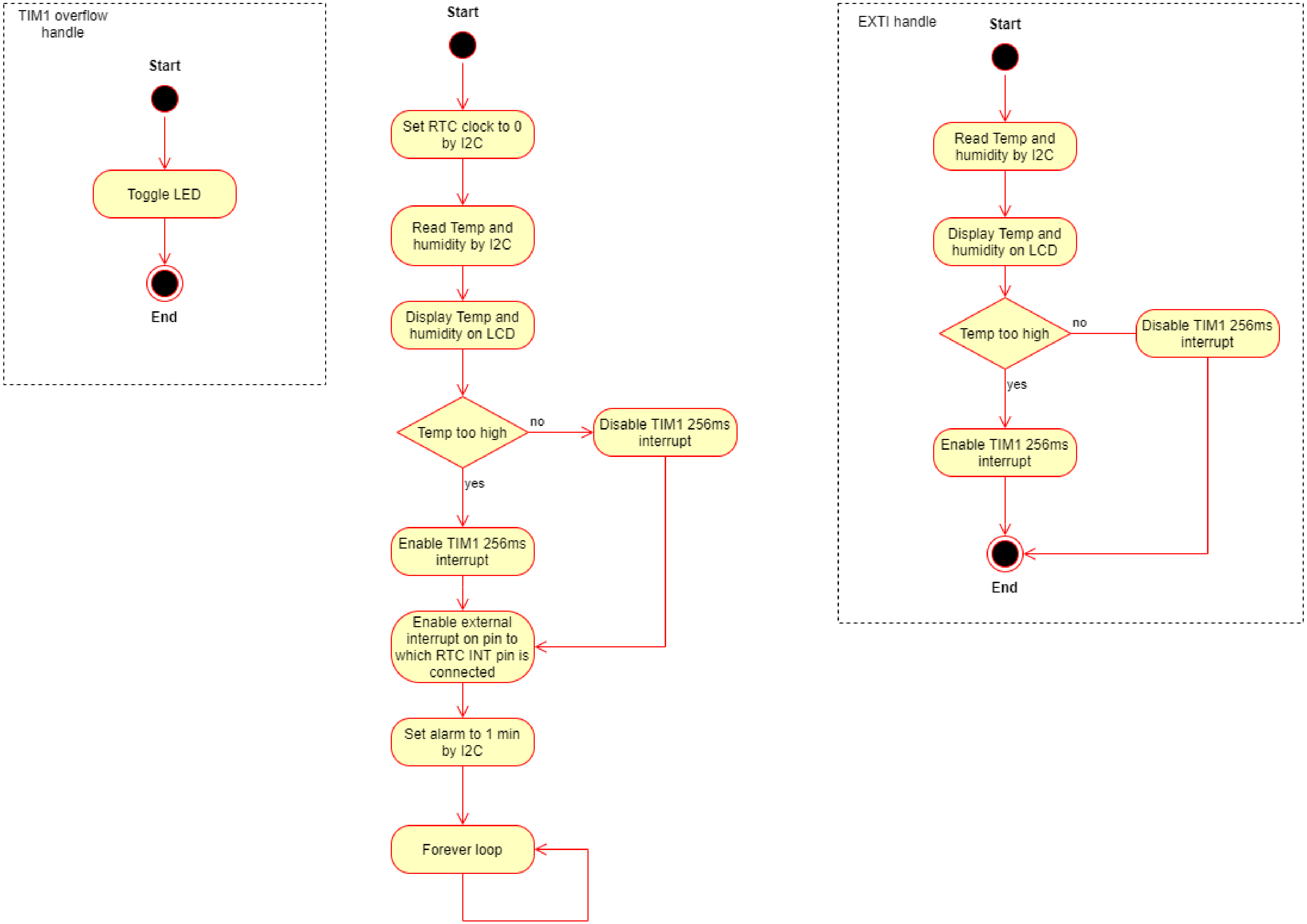


→ ALL ACK/NACK done by SLAVE

Meteo station

Consider an application for temperature and humidity measurement and display. Use combine sensor DHT12, real time clock DS3231, LCD, and one LED. Application display time in hours:minutes:seconds at LCD, measures both temperature and humidity values once per minut, display both values on LCD, and when the temperature is too high, the LED starts blinking.

1. FSM state diagram picture of meteo station. The image can be drawn on a computer or by hand. Concise name of individual states and describe the transitions between them.


TIM1 overflow handle
Start
Toggle LED
End
Start
Set RTC clock to 0 by I2C
Read Temp and humidity by I2C
Display Temp and humidity on LCD
Temp too high
no
Disable TIM1 256ms interrupt
yes
Enable TIM1 256ms interrupt
Enable external interrupt on pin to which RTC INT pin is connected
Set alarm to 1 min by I2C
Forever loop
EXTI handle
Start
Read Temp and humidity by I2C
Display Temp and humidity on LCD
Temp too high
no
Disable TIM1 256ms interrupt
yes
Enable TIM1 256ms interrupt
End