Dynamic color changer

1. Authors

- **Kamil Sokołowski**
- AGH University of Science and Technology, The Faculty of Computer Science, Electronics and Telecommunications
- Department of Electronics
- ksokolowski95@gmail.com

2. Basic aim

To make simple application which will connect microprocessor with peripheral sensors trough I2C bus and convert gathered data to useful format.

3. How to use

Connect microcontroller to power supply and UART->USB adapter to USB port of your PC. Determine at which port is assigned adapter. Launch **SMPProject.exe** application, choose right COM port and click ,,connect” button.

4. Used hardware

- Freedom KL46Z
  - ARM ® Cortex M0+
  - 256 KB flash
- FT232R USB UART IC
  - USB to serial UART interface
- MPU-6050
  - Six-Axis Gyro + Accelerometer

5. Programming platforms

- Keil uVision 5
  - Programming FRDM-KL46z
- Visual Studio 2015
  - Converting raw data and creating front end application

6. Used communication modules

- I2C

  Most significant features:
  - Only two bus lines are required
  - No strict baud rate requirements, master device generates bus clock on SCL line
  - Simple master/slave relationship between connected device
  - Up to 127 connected devices to one bus (7-bit address + 1 R/W bit)
  - Transmission speed up to 3,4 Mbit/s

Example:

I^2C bus consist of two signals: SCL(clock) and SDA(data). SCL is always generated by master. The I^2C bus drivers are open drain type, thus slave can only pull it down.

Message is separated into two types frames: address and data.

Transmission:

- Master leaves SCL on high and pulls SDA low, this is the point of start
- Master send address of slave to communicate. 7 bit address + 1 R/W bit
- If connection was successful slave device should take control over SDA line and pull it down, in other case, the connection is not initialized properly
- If ACK bit = '1' device, either the master or slave, depending in value of R/W bit, can put data on SDA
- Once all the data was send, master will generate stop signal, which is defined by 0->1 on SDA, right after 0->1 on SCL
- Exchange of several messages is allowed by restart condition
  - SDA goes high, when SCL is low, then SCL goes high, SDA goes low again (SCL still high), in effect, next message can be send

- UART

  UART is asynchronous serial communication, where data format and transmission speeds are configurable

  Typical UART frame is 8 bit + start and stop bits. In addition, we can add Parity bit and one more stop bit.

  UART is simple and old form of data transmitting. The most important thing is to set properly set both, receiver and transmitter. They must be initialized with the same BR value, due to locally generated clock signal. Significant case is also crossing RX/TX line when connecting 2 UART modules (e.g connection between UART module from microcontroller and UART->USB adapter).

7. Function description

a. Embedded system

- I2C
  I2C_Type *I2C_module – pointer to choose I2C module to use
  - I2C1_init(void) – initialization of I2C1 module
  - I2C2_init(void) – initialization of I2C2 module
  - I2C_enable(I2C_Type *I2C_module) – enable I2C module
  - I2C_enable(I2C_Type *I2C_module) – disable I2C module
  - I2C_mode_TX(I2C_Type *I2C_module) – I2C mode: Transmitter
  - I2C_mode_TX(I2C_Type *I2C_module) – I2C mode: Receiver
  - I2C_start(I2C_Type *I2C_module) – sending *start signal* to start transmission
  - I2C_stop(I2C_Type *I2C_module) – sending *stop signal* to stop transmission
  - I2C_restart(I2C_Type *I2C_module) – sending *restart signal* to restart transmission
  - I2C_ack(I2C_Type *I2C_module) - sending *ack signal* to acknowledgment transmission
  - I2C_nack(I2C_Type *I2C_module) - sending n*ack signal* to not acknowledgment transmission
  - I2C_check_ACK(I2C_Type *I2C_module) - checking slave device answer, ack or nack
  - I2C_send(I2C_Type *I2C_module, uint8_t data) – sending one byte of data
  - I2C_read(I2C_Type *I2C_module) – reading one byte of data
  - I2C_wait(I2C_Type *I2C_module) – waiting for end of transmission
  - I2C_read_byte(I2C_Type *I2C_module, uint8_t device_address, uint8_t read_register) – complete algorithm to read data from slave device register
  - I2C_write_byte(I2C_Type *I2C_module, uint8_t device_write_address, uint8_t write_register, uint8_t data) – complete algorithm to write data to slave device register
- UART
  UART_Type *UART_module – pointer to choose UART module to use
  - UART1_init(uint32_t BAUD_RATE) - initialization of UART1 module with specified baudrate
  - UART2_init(uint32_t BAUD_RATE) - initialization of UART2 module with specified baudrate
  - UART_send(UART_Type *UART_module,char *string) – sending data string
  - UART_send_int(UART_Type *UART_module, int count) – sending int value

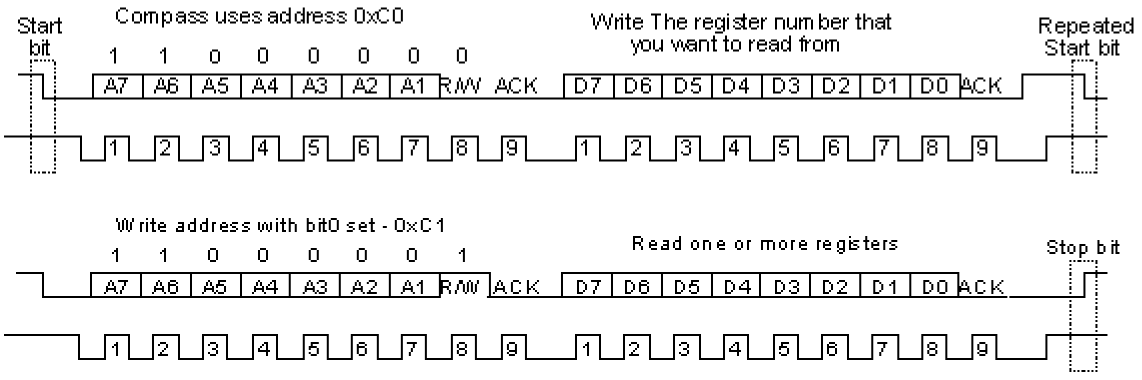
- MPU-6050
  - MPU6050_init(void) – initialization of MPU-6050 module
  - MPU6050_read_byte(uint8_t read_register) – reading one byte of data from MPU-6050
  - MPU6050_write_byte(uint8_t write_register, uint8_t data) - writing one byte of data to MPU-6050
  - MPU6050_get_raw_accel_data(int16_t *raw_data) – reading MPU-6050 accelerometer measurements

b. Universal App C#

- uartPort_DataReceviedHandler(object sender, SerialDataReceviedEventArgs e) – even handler when data is received
- RawData_ToIntArray() – converting data from UART to array of integers
- ConvertedAccelData_ToRGB() – scaling measurements to range of RGB
- SetPictureBoxColor() – setting color of picture box depending on accelerometer values

ADDITIONAL INFO:

I2C bit sequence


Start bit
Compass uses address 0xC0
1 1 0 0 0 0 0 0
A7 A6 A5 A4 A3 A2 A1 R/W ACK
Write The register number that you want to read from
D7 D6 D5 D4 D3 D2 D1 D0 ACK
Repeated Start bit
1 2 3 4 5 6 7 8 9
Write address with bit0 set - 0xC1
1 1 0 0 0 0 0 1
Read one or more registers
Stop bit