

UNIT 13
THE I2C BUS

1

Aim and Agenda of unit 12

The aim of the presentation
Give the basic ideas and simple examples for enable Arduino to communicate with any kind of device or peripheral

The agenda of the presentation

- Explain what is **Communication** and especially Serial Communication
- Present **Programming Communication Functions** used in this unit (basic principles)
- Present **Programming Transmitting Data Functions** used in this unit (basic principles)
- Present **Programming Receiving Data Functions** used in this unit (basic principles)
- Present **Other General Purpose Functions**
- Practice Section

2

Introduction

- **Current controllers** have circuits for bit by bit serial communications
 - USART
 - SPI
- **Communication Protocols**
 - I2C
 - 2-wire Interface

3

I²C Protocol

- Developed by Philips in the 90's to connect up integrated circuits within single devices or household appliances
- These days it's universal
- *More information:*
http://www.nxp.com/products/interface_and_connectivity/i2c/

Co-funded by the Erasmus+ Programme

open In

4

THE I2C BUS

- *Communication system that transfers data between components inside a computer or between computers.*
- *Uses only two signals or pins to exchange information*

NAME	DESCRIPTION
SCL	Serial clock line. It's always output when used for the master or host and input for the slave.
SDA	Serial data line (bidirectional). The data can travel from the master to the slave or the other way around.

Co-funded by the Erasmus+ Programme

open In

5

THE I2C BUS - FEATURES

- *It uses just two signals to transfer data: SCL and SDA.*
- *They're both open collector signals so they have to be connected to +V using two separate pull-up resistors.*
- *Byte wise data transfer. 8 bits is the minimum for each word transferred.*
- *"Multi-master" replication system. A single bus (SCL, SDA) may include several master or host controllers as well as several slaves.*
- *All slave devices have an address assigned them during their manufacture; it differentiates them from other slaves on the bus.*
- *A single bus cannot accommodate two slave devices with the same address.*
- *The master or host uses this address to select the slave device it wants to communicate with.*

Co-funded by the Erasmus+ Programme

open In

6

THE I2C BUS - TERMINOLOGY

- **Bit transfer:**
 - ✓ *I²C protocol uses synchronous communication.*

SDA

SCL

bit 1 or bit 0

bit 1 or bit 0

Valid bit

Non-valid bit

Co-funded by the Erasmus+ Programme

open In

7

THE I2C BUS - TERMINOLOGY

- **Start/Stop Condition:**
 - ✓ *The host initiates all transfers by sending a sequence or Start (S) condition and ends transfers with a Stop (P) condition.*

SDA

SCL

S

P

Co-funded by the Erasmus+ Programme

open In

8

THE I2C BUS - TERMINOLOGY

- **The Acknowledge Bit:**
 - ✓ *No limitation on the number of bytes however each one must be followed by an Acknowledge bit or ACK/NACK.*

Transmitter SDA

Receiver SDA

Master SDA

Negative acknowledgment

Acknowledgment

1 2 ... 8 9

↑

Clock tick for ACK acknowledgment bit

Co-funded by the Erasmus+ Programme

open In

9

THE I2C BUS - TERMINOLOGY

- **The I2C Transaction Frame:**
 - ✓ All I2C transfers consist of a frame of one or more bytes.

INICIO DIRECCION R/W ACK DATO ACK DATO ACK STOP

Co-funded by the Erasmus+ Programme

open In

10

THE I2C BUS - WIRE LIBRARY

- Functions contained in this library manage the electronic circuits or hardware integrated in the Arduino controller to implement what we called "SSP" or Synchronous Serial. This library comes with the Arduino Integrated Development Environment (IDE)
- **The Wire.begin() FUNCTION**
 - **Syntax:** `Wire.begin(address)`
 - ✓ *address:* Optional seven bit integer (between 0 and 128). If it isn't indicated it is assumed that the device, in this case the Arduino UNO, will function as the master or host. Otherwise it is assumed that the Arduino UNO will function as a slave
 - On the Arduino UNO board pins A4 and A5 become SDA and SCL signals respectively and you must not connect any other sort of device to them

Co-funded by the Erasmus+ Programme

open In

11

THE I2C BUS - WIRE LIBRARY

- **The Wire.beginTransmission() FUNCTION**
 - **Syntax:** `Wire.beginTransmission(address)`
 - ✓ *address:* A seven bit integer (between 0 and 128) which represents the address of the device the slave wants to talk to.
- **The Wire.write() FUNCTION**
 - **Syntax:** `Wire.write(value)`
 - **Syntax:** `Wire.write(string)`
 - **Syntax:** `Wire.write(data,length)`
 - ✓ *value:* a value to send as a single byte or eight bits
 - ✓ *string:* a chain of characters comprising several bytes
 - ✓ *data:* an array of data to be sent as bytes
 - ✓ *length:* the number of bytes to transmit

Co-funded by the Erasmus+ Programme

open In

12

THE I2C BUS - WIRE LIBRARY

▪ The `Wire.endTransmission()` FUNCTION

- **Syntax:** `Wire.endTransmission(mode)`
 - ✓ *mode*: `TRUE` or `FALSE`. If `true`, `endTransmission()` sends a stop message after transmission, releasing the I2C bus. If `false`, `endTransmission()` sends a restart message (S) after transmission. This message is required by some I2C devices and is optional. The default value is `true`.
- **Returns:** The function returns the following error codes for assessment:
 - ✓ 0: success
 - ✓ 1: data too long to fit in transmit buffer
 - ✓ 2: received NACK on transmit of address
 - ✓ 3: received NACK on transmit of data
 - ✓ 4: other error

Co-funded by the
Erasmus+ Programme

13



13

THE I2C BUS - WIRE LIBRARY

▪ The `Wire.requestFrom()` FUNCTION

- **Syntax:** `Wire.requestFrom(address, quantity, mode)`
 - ✓ *address*: a seven-bit integer (between 0 and 128) and represents the address of the device which bytes will be requested from.
 - ✓ *quantity*: number of bytes to be received.
 - ✓ *mode*: `TRUE` or `FALSE`. `TRUE` will send a stop message (P) after the request and reception of all the bytes, releasing the bus. `FALSE` will send a constant restart (S) after the request thus keeping the connection active. This message is required by some I2C devices and is optional. The default value is `true`.

Co-funded by the
Erasmus+ Programme

14



14

THE I2C BUS - WIRE LIBRARY

▪ The `Wire.available()` FUNCTION

- **Syntax:** `Wire.available()`

▪ The `Wire.read()` FUNCTION

- **Syntax:** `Wire.read()`.

▪ The `Wire.onReceive ()` FUNCTION

- **Syntax:** `Wire.onReceive(function)`.
 - ✓ *function*: registers the function to be called when a slave device receives a transmission from a master. This function usually reads the bytes that the master writes to it.

▪ The `Wire.onRequest ()` FUNCTION

- **Syntax:** `Wire.onRequest()`.
 - ✓ *handler*: function to be called every time the master requests data from it.

Co-funded by the
Erasmus+ Programme

15



15

THE I2C BUS - WIRE LIBRARY

- **The Wire.available() FUNCTION**
 - **Syntax:** `Wire.available()`
- **The Wire.read() FUNCTION**
 - **Syntax:** `Wire.read()`.
- **The Wire.onReceive () FUNCTION**
 - **Syntax:** `Wire.onReceive(function)`.
 - ✓ *function*: registers the function to be called when a slave device receives a transmission from a master. This function usually reads the bytes that the master writes to it.
- **The Wire.onRequest () FUNCTION**
 - **Syntax:** `Wire.onRequest()`.
 - ✓ *handler*: function to be called every time the master requests data from it.

Co-funded by the Erasmus+ Programme open In

16

THE I2C BUS - WIRE LIBRARY

- **Before start:**
 - *Take a look of the examples that exist in this library:*
 - ✓ *master_writer*: configures the Arduino as an I2C master device for transmitting data.
 - ✓ *slave_receiver*: configures the Arduino as a slave data receiver.
 - ✓ *master_reader*: configures the Arduino as a master data receiver.
 - ✓ *slave_sender*: configures the Arduino as an I2C master device for transmitting data.

Co-funded by the Erasmus+ Programme open In

17

THE I2C BUS - DEVICES

- **THE SRF02 ULTRASONIC RANGE FINDER**
 - **Operational Principles**
 - ✓ From 20 KHz onwards.
 - ✓ A capsule emits an ultrasonic signal. This bounces off an object and creates an echo that goes back to the range finder. It then measures the time transpired between the signal and the echo.
 - ✓ Ultrasonic waves move at the speed of sound: 343 m/s through the air at sea level at a temperature of 20°C and relative humidity of 50%.

Distance	Time	Description
1 cm	$0.00002915 \text{ " } = 0.02915 \text{ mS } = 29.15 \text{ }\mu\text{S}$	$(1 / 343) / 100$
1 m	$0.002915 \text{ " } = 2.915 \text{ mS } = 2915 \text{ }\mu\text{S}$	$1 / 343$
1 Km	$2.915 \text{ " } = 2915 \text{ mS } = 2915000 \text{ }\mu\text{S}$	$(1 / 343) * 1000$

Co-funded by the Erasmus+ Programme open In

18

THE I2C BUS - DEVICES

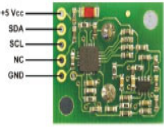
- **SRF02 - FEATURES AND CONNECTIONS**
 - **important features of the SRF02 ultrasonic rangefinder:**
 - ✓ Range: from 16 cm to 6 m (in theory)
 - ✓ Power: + 5 V a 4 mA
 - ✓ Ultrasonic frequency: 40 MHz
 - ✓ Size: 24 mm x 20 mm x 17 mm
 - ✓ Analogue Gain: Automatic 64 step gain control
 - ✓ Connection Modes: 1 - Standard I2C Bus 2 - Serial Bus (connects up to 16 devices to any UART serial port)
 - ✓ Full Automatic Tuning: No calibration, just power up and go
 - ✓ Units: Range reported in μ S, mm or inches.
 - ✓ The default shipped address of the SRF02 is 224 (0xE0). It can be changed by the user to any of 16 addresses: E0, E2, E4, E6, E8, EA, EC, EE, F0, F2, F4, F6, F8, FA, FC or FE, therefore up to 16 different addresses can be used.

Co-funded by the Erasmus+ Programme open In

19

THE I2C BUS - DEVICES

- **SRF02 - FEATURES AND CONNECTIONS**
 - *The SRF02 functions as an I2C slave device and includes its own controller responsible for making measurements and calibrations and then transmitting them to the host controller*



Pin	Name	
1	+5v Vcc	Voltage 5 V
2	SDA	I2C bus data line
3	SCL	I2C time line
4	Mode	Left unconnected
5	GND	Ground

Co-funded by the Erasmus+ Programme open In

20

THE I2C BUS - DEVICES

- **SRF02 - INTERNAL REGISTERS**
 - *A summary of the internal registers of the SRF02 appear in the table below:*

Location	Read	Write
0	INTERNAL VERSION OF I2C DEVICE FIRMWARE	COMMAND REGISTER
1	UNUSED (READS 0X80)	N/A
2	RANGE HIGH BYTE	N/A
3	RANGE LOW BYTE	N/A
4	AUTOTUNE MINIMUM - HIGH BYTE	N/A
5	AUTOTUNE MINIMUM - LOW BYTE	N/A

Co-funded by the Erasmus+ Programme open In

21

THE I2C BUS - DEVICES

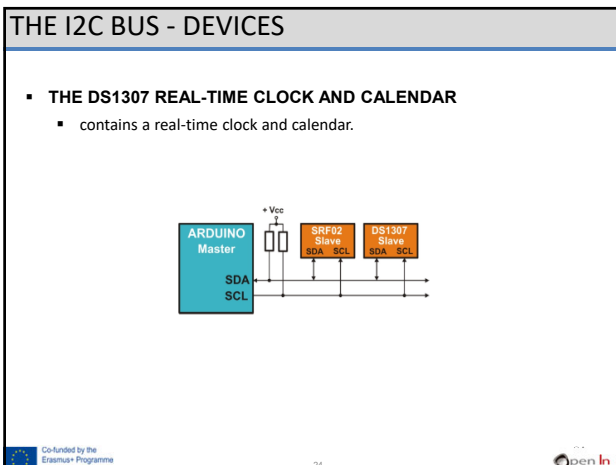
Command		Action
Decimal	Hex	
80	0X50	REAL RANGING MODE - RESULT IN INCHES
81	0X51	REAL RANGING MODE - RESULT IN CENTIMETRES
82	0x52	REAL RANGING MODE - RESULT IN MICRO-SECONDS
86	0X56	FAKE RANGING MODE - RESULT IN INCHES
87	0x57	FAKE RANGING MODE - RESULT IN CENTIMETRES
88	0x58	FAKE RANGING MODE - RESULT IN MICRO-SECONDS
92	0x5C	TRANSMIT AN 8 CYCLE 40KHZ BURST - NO RANGING TAKES PLACE
96	0X60	FORCE AUTOTUNE RESTART - SAME AS POWER-UP. YOU CAN IGNORE THIS COMMAND
160	0xA0	1ST IN SEQUENCE TO CHANGE I2C ADDRESS
165	0xA5	3RD IN SEQUENCE TO CHANGE I2C ADDRESS
170	0xAA	2ND IN SEQUENCE TO CHANGE I2C ADDRESS

Co-funded by the Erasmus+ Programme 22

22

- ### THE I2C BUS - DEVICES
- **Real ranging:**
 - ✓ measures the distance of an object from the SRF02. The device emits an 8 cycle ultrasonic burst at 40 KHz. It then waits to receive an echo - if there is one, that is.
 - **Fake ranging:**
 - ✓ xxx
 - **Burst:**
 - ✓ Bursts don't perform any measurement. It's used as a warning signal or a synchroniser in an environment where there are several SRF02 sensors.
 - **Restart**
 - ✓ it performs the initial tasks of adjustment and calibration.
- Co-funded by the Erasmus+ Programme 23

23



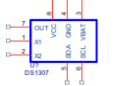
24

THE I2C BUS - DEVICES

- THE DS1307 REAL-TIME CLOCK AND CALENDAR
 - Features:
 - ✓ It contains a real-time clock and calendar. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information.
 - ✓ It includes corrections for leap years to the year 2099.
 - ✓ It contains 56 bytes of NV SRAM or Non volatile RAM memory powered by an external Non volatile battery.
 - ✓ I2C interface with the host or master.
 - ✓ Programmable square-wave output signal.
 - ✓ The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply.
 - ✓ Low power operation extends battery backup run time and consumes less than 500nA in battery backup mode with oscillator
 - ✓ 8-Pin DIP and 8-Pin SO minimizes required space.

25

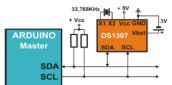
THE I2C BUS - DEVICES

- THE DS1307 REAL-TIME CLOCK AND CALENDAR
 - Enclosure and PIN-OUT:
 

Nº	Name	Description
1	X1	the input to the oscillator and is connected to an external quartz crystal 32.768kHz oscillator.
2	X2	the output to the oscillator and is connected to an external quartz crystal 32.768kHz oscillator.
3	VBAT	Backup supply input for any standard 3V lithium cell or other energy source.
4	GND	The earth for the main power supply
5	SDA	Serial Data Input/Output. Input/output for the I2 C serial interface.
6	SCL	Serial Clock Input. SCL is the clock input for the I2C interface
7	SQWE/OUT	Square wave/output driver.
8	VCC	Primary +5V power supply.

26

THE I2C BUS - DEVICES

- THE DS1307 REAL-TIME CLOCK AND CALENDAR
 - Internal Registers:
 

27

THE I2C BUS - DEVICES

- **THE DS1307 THE DS1307 LIBRARY**
 - **The getRegister() FUNCTION:**
 - **Syntax:** *getRegister(n)*.
 - ✓ *n*: the number of the internal register of the DS1307 to be read (between 0 and 63).
 - **The resume() FUNCTION:**
 - **Syntax:** *resume ()*.
 - **The standby() FUNCTION:**
 - **Syntax:** *standby ()*.

Co-funded by the Erasmus+ Programme 34

34

THE I2C BUS - DEVICES

- **THE DS1307 THE DS1307 LIBRARY**
 - **The getDate() FUNCTION:**
 - **Syntax:** *getDate(buffer)*.
 - ✓ *buffer*: seven position byte array that carries the date and time contained in the DS1307:

Buffer	Description
Buffer[0]	seconds (0-59)
Buffer[1]	minutes (0-59)
Buffer[2]	hours (1-12 or 0-23)
Buffer[3]	day of the week (1-7)
Buffer[4]	day of the month (1-31)
Buffer[5]	month (1-12)
Buffer[6]	year (0-99)

Co-funded by the Erasmus+ Programme 35

35

THE I2C BUS - DEVICES

- **THE DS1307 THE DS1307 LIBRARY**
 - **The setSeconds() FUNCTION:**
 - **Syntax:** *setSeconds(v)*.
 - ✓ *v*: the new value for the seconds between 0 and 59.
 - **The setMinutes() FUNCTION:**
 - **Syntax:** *setMinutes(v)*.
 - ✓ *v*: the new value for the minutes between 0 and 59.
 - **The setHours() FUNCTION:**
 - **Syntax:** *setHours(v)*.
 - ✓ *v*: the new value for the hours between 1 and 12 or 0 and 23.

Co-funded by the Erasmus+ Programme 36

36

THE I2C BUS - DEVICES

- **THE DS1307 THE DS1307 LIBRARY**
 - **The setDow() FUNCTION:**
 - **Syntax:** *setDow(v)*.
 - ✓ *v*: the new value for the day of the week between 1 and 7.
 - **The setData() FUNCTION:**
 - **Syntax:** *setData(v)*.
 - ✓ *v*: the new value for the day of the month between 1 and 31.
 - **The setMonth () FUNCTION:**
 - **Syntax:** *setMonth(v)*.
 - ✓ *v*: the new value for the month between 1 and 12.
 - **The setYear() FUNCTION:**
 - **Syntax:** *setYear(v)*.
 - ✓ *v*: the new value for the day of the week between 0 and 99.

37

THE I2C BUS – PRACTICE SECTION

- **EXAMPLE 1: A VERSION OF FIRMWARE**
 - The version of firmware that controls the SRF02 ultra sonic range finder is displayed on the LCD.

38

THE I2C BUS – PRACTICE SECTION

- **EXAMPLE 1: A VERSION OF FIRMWARE**
 - The A5 and A4 analogue inputs also behave like SCL and SCD signals respectively on the Arduino UNO I2C bus. They're connected to the R3 and R4 pull-up resistors and the SCL and SDA pins on the SRF02 ultra sonic range finder

```

void loop ()
{
  // STEP 1
  Wire.beginTransmission(112);
  Wire.write(byte(0));
  Wire.endTransmission();

  // STEP 2
  Wire.requestFrom(112,1);

  // STEP 3
  if(Wire.available())
    ver = Wire.read();

  lcd_print(ver);
  while(1);
}
    
```

39

THE I2C BUS – PRACTICE SECTION

- **EXAMPLE 2: DISTANCE**
 - The screen will display the distance in centimetres between the range finder and an object.
 - ✓ *Step 1:* Communication with the SRF02 Slave is enabled by selecting the command register No. 0 and writing the value 81. This command initiates a new measurement of the distance in centimetres.
 - ✓ *Step 2:* The waiting cycle. The SRF02 requires a minimum of 65 mS to complete the measurement.
 - ✓ *Step 3:* Select the first register and save the result of the measurement. The most significant byte is stored in register No. 2 and the least significant one in No. 3.

Co-funded by the Erasmus+ Programme 40

40

THE I2C BUS – PRACTICE SECTION

- **EXAMPLE 3: MORE DISTANCES**
 - The SRF02 will provide results in centimeters, inches and microseconds and they'll be displayed on the LCD screen.
 - ✓ *medir(cm):* Performs the measurement in centimeters.
 - ✓ *medir(in):* Performs the measurement in inches.
 - ✓ *medir(uS):* Performs the measurement in microseconds.

Have close look at this extract from a program:

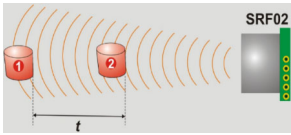
```
measure(in);
lcd.setCursor(13,0);
n=lcd.print(distance);
for(n; n<3; n++)
  lcd.print(" "); //Complete with blank space
```

Co-funded by the Erasmus+ Programme 41

41

THE I2C BUS – PRACTICE SECTION

- **EXAMPLE 4: COLLISION AVOIDANCE SYSTEM**
 - Detect the distance between the SRF02 ultrasonic range finder and an object or obstacle. As the object gets closer and closer to the SRF02 and passes certain minimum distances the piezo-electric buzzer connected to the D13 output pin starts to emit a warning signal at different frequencies.
- **EXAMPLE 5: SPEED DETECTOR**
 - This exercise is purely experimental and while it's not easy to check its accuracy it can give you a basic idea of what a speed detector is like.



Co-funded by the Erasmus+ Programme 42

42

THE I2C BUS – PRACTICE SECTION

- EXAMPLE 6: AREA METRE**
 - These metres are able to use ultrasound or laser light to measure distances. In this case, of course we'll be using the SRF02 ultrasonic range finder.

Co-funded by the Erasmus+ Programme

43

open ln

43

THE I2C BUS – PRACTICE SECTION

- EXAMPLE 7: REAL-TIME CLOCK**
 - read the first seven internal registers to get the hours, minutes and seconds and then display them in real time.

Co-funded by the Erasmus+ Programme

44

open ln

X1	1	8	Vcc
X2	2	7	SDI/O
Vcc	3	6	SCL
GND	4	5	SDA

44

THE I2C BUS – PRACTICE SECTION

- EXAMPLE 8: CLOCK AND CALENDAR PART 1**
 - This exercise is just like a continuation of the previous one: the date and the time are displayed on the LCD screen. Two new functions have been created to make this possible; you'll be able to use them in future projects too: visuDate() and visuTime().

Co-funded by the Erasmus+ Programme

45

open ln

45

THE I2C BUS – PRACTICE SECTION

- EXAMPLE 9: CLOCK AND CALENDAR PART 2**
 - This example uses the previous visuDate() and visuTime() functions to run a real-time clock and calendar; you can also adjust any of the following fields: day of the week, day of the month, month, year, hours and minutes.

Co-funded by the Erasmus+ Programme of the European Union

46

46

THE I2C BUS – PRACTICE SECTION

- EXAMPLE 10: BILLBOARD**
 - You're going to get into the DS1307 real-time clock and calendar to get the date and the time out by using the I2C protocol. You're also going to communicate with the DHT11 sensor from the previous unit to get the room temperature and the relative humidity by using the 1-wire protocol.
- EXAMPLE 11: BILLBOARD**
 - It sends a record of the date, time, humidity and temperature at regular intervals using serial communication.

Co-funded by the Erasmus+ Programme of the European Union

47

47

Co-funded by the Erasmus+ Programme of the European Union

UNIT 13 THE I2C BUS
Thank You

48