SMART NANNY

RAPORT & DOCUMENTATION

REPORT

Thursday, 10:00

Date: 18.04.2019

Team:

Major:

Information Technology

IFE, 4th semester

Members:

Michał Suliborski *(leader)*	217863
Emilia Markowska	217854
Jędrzej Szor	217864
Arkadiusz Zasina	217868

Tasks assignment

Michał Suliborski *(leader)*	Wireless communication Devices assembly
Emilia Markowska	Buzzer Sound detection
Jędrzej Szor	RTC Clock Code maintenance
Arkadiusz Zasina	Humidity and temperature detection LCD configuration

Devices used

- 2 x Arduino - Uno R3,
- 2 x Wireless Radio Module - nRF24L01,
- 1 x LCD KeyPad Shield,
- 1 x Temperature and Humidity Sensor - DHT11,
- 1 x RTC - DS1307,
- 1 x Sound Sensor Module - VM309
- 1 x Piezo buzzer

1. Project description

1.1 General description

The project is an embedded electronic assistant, which is called “Smart Nanny”, which aim is to make childcare easier and safer. It monitors baby’s behaviour and environmental conditions in which the baby is present. It sends those data to external device and alarms in case of baby cry or bad environment conditions. The whole project is a set of two, connected wirelessly devices.

1.2 Data measuring device

First device is the condition and behaviour monitor. It ought to be situated in child’s surrounding. Using humidity and temperature sensor it gathers information about surrounding as well as detects cry using sound sensor. Those information are wirelessly transmitted to second device.

1.3 Control device

This device is the core of the whole project. It is equipped with LCD screen, which allows it to display function menu. The device receives information from monitor device periodically and displays them as well as current date and time. In addition it allows user to set custom alarms and buzz in case of detecting baby cry or bad environment conditions.

1.4 User guide

The emitter part should be placed up to 1.5m from child according to the test in order to precisely measure temperature, humidity and baby’s voice level.

The receiver part should be under parent’s control. Receiver is equipped with LCD screen with buttons, with which user may navigate the menu. Menu has following options embedded:

- setting and deleting alarms - once alarm is set, it will be activated every day in a given hour until it is deleted by the user,
- monitoring current temperature and humidity in baby’s room
- giving current time and date,
- informing about potential dangers - message displayed on the screen and sound signal from buzzer in case of situations such as: too high or too low temperature, baby’s crying.

2. Peripherals and interface configuration

2.1 GPIO

Table below presents the pinout of emitter element:

Pin	Alias	PWM	Assignment
0	RX	NO	not used
1	TX	NO	not used
2	INT0	NO	not used
3	INT1	YES	not used
4	-	NO	not used
5	-	YES	not used
6	-	YES	not used
7	-	NO	Temperature and Humidity Sensor DHT11
8	-	NO	Wireless Radio Arduino nRF24L01
9	-	YES	Wireless Radio Arduino nRF24L01
10	SS	YES	not used
11	MOSI	YES	Wireless Radio Arduino nRF24L01
12	MISO	NO	Wireless Radio Arduino nRF24L01
13	SCK	NO	Wireless Radio Arduino nRF24L01
14	A0	N/A	not used
15	A1	N/A	not used
16	A2	N/A	not used
17	A3	N/A	not used
18	A4, SDA	N/A	not used
19	A5, SCL	N/A	Sound Sensor Module VM309

Table below presents the pinout of receiver element:

Pin	Alias	PWM	Assignment
0	RX	NO	LCD KeyPad Shield
1	TX	NO	Piezo buzzer
2	INT0	NO	Wireless Radio Arduino nRF24L01
3	INT1	YES	Wireless Radio Arduino nRF24L01
4	-	NO	LCD KeyPad Shield
5	-	YES	LCD KeyPad Shield
6	-	YES	LCD KeyPad Shield
7	-	NO	LCD KeyPad Shield
8	-	NO	LCD KeyPad Shield
9	-	YES	LCD KeyPad Shield
10	SS	YES	Wireless Radio Arduino nRF24L01
11	MOSI	YES	Wireless Radio Arduino nRF24L01
12	MISO	NO	Wireless Radio Arduino nRF24L01
13	SCK	NO	Wireless Radio Arduino nRF24L01
14	A0	N/A	Keys of LCD KeyPad Shield
15	A1	N/A	not used
16	A2	N/A	not used
17	A3	N/A	not used
18	A4, SDA	N/A	RTC DS1307 I2C
19	A5, SCL	N/A	RTC DS1307 I2C

2.2 I2C

I2C is a serial bidirectional bus used for sending data. It consists of two pins: SDA(Serial Data Line) and SCL(Serial Clock Line). The data flow is synchronized by the clock signal. ATmega 328p MCU refers to it as TWI(Two Wire Interface). The transmission begins by resetting TWINT flag. Later, the TWSR value is checked to verify if the Start condition was successfully transmitted. Then the application loads SLA+W to TWDR and when the appropriate value is written to TWCR, the SLA+W in TWDR is transmitted. If SLA+W was successfully sent and acknowledged, the application again resets TWINT and loads actual Data to TWDR and control signals to TWCR. Data is then sent and if ACK was received and value in TWSR checks out, application resets TWINT and loads appropriate Stop signal to TWCR.

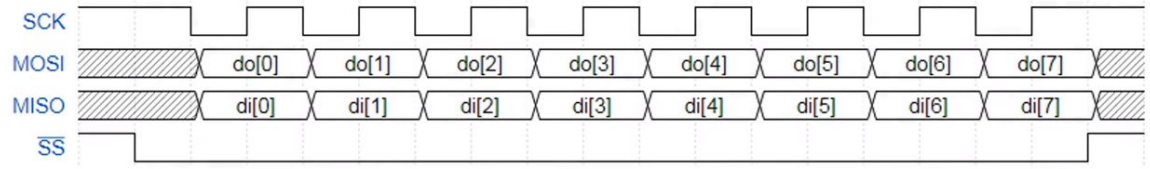
2.3 SPI

The Serial Peripheral Interface allows high-speed synchronous data transfer between the ATmega328p and the nRF24 wireless module. We use this interface to enable wireless communication between two devices.

Serial Peripheral Interface works in Master-Slave manner and uses 4 pins.

- SCLK/SCK - Serial clock
- MOSI - Master out, Slave in (also known as SDI - Send Data In)
- MISO - Master in, Slave out (also known as SDO - Send Data Out)
- SS - Slave Select (also known as Chip Select)

As presented below, the SPI Master initiates the communication cycle when pulling low on the Slave Select pin of the desired Slave. Master and Slave prepare the data to be sent in their respective shift registers, and the Master generates the required clock pulses on the SCK line to interchange data.
Data is always transferred from Master to Slave on the MOSI line, and from Slave to Master on the MISO line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select line.



2.4 Temperature/humidity detector - DHT11

DHT11 module is used to gather information about temperature and the level of humidity. To communicate with MCU, it uses single-bus data format. Data transferred is divided into parts consisting of 40 bits, 8 of which is dedicated to check sum. Rest is divided into two parts for integral and decimal data. The data is sent in equal intervals of 3 seconds. When MCU sends a start signal, DHT11 changes from the low-power-consumption mode to the running-mode, waiting for MCU completing the start signal. Once it is completed, DHT11 sends a response signal that include the relative humidity and temperature information to MCU. Once data is collected, DHT11 will change to the low-power-consumption mode until it receives a start signal from MCU again.
In order to use DHT11 module, we included a library used for converting the data into double values:

```
#include <dht.h>
```

Data collection along with its conversion is achieved with:

```
DHT.read11(DHT11_PIN)
```

followed by:

```
DHT.temperature
DHT.humidity
```

2.5 LCD Shield

LCD Keypad Shield is a module that provides an interface for going through a menu and select one of available options. It consists of blue backlight LCD able to display 1602 white characters as well as 5 buttons keypad. The keypad uses single ADC channel. 5 stage voltage divider is used in order to read key value. It uses D4 - D7 pins for data transmission, D8 for register selection, D9 as enable pin and D10 for backlight control. Display allows to simultaneously show two rows 16 characters each. First we created the library object providing subsequent pins:

```
#include "LiquidCrystal.h"
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
```

Then we had to declare the layout of our screen:

```
lcd.begin(16, 2);          //2 lines, 16 characters each
```

The keyboard is monitored by reading just one analog pin:

```
int read_LCD_buttons() {
    adc_key_in = analogRead(0);
    if (adc_key_in > 1000) return btnNONE;
    if (adc_key_in < 50) return btnRIGHT;
    if (adc_key_in < 250) return btnUP;
    if (adc_key_in < 450) return btnDOWN;
    if (adc_key_in < 650) return btnLEFT;
    if (adc_key_in < 850) return btnSELECT;
    return btnNONE;
}
```

To control what is shown, we used functions lcd.print() combined with lcd.setCursor() and lcd.clear().

2.6 Sound detector - VMA309

Sound detector which is installed in our device, uses GPIO interface embedded in Arduino. It modifies received voltage and converts it to the number in the range from 0 to 1023. We assigned the value of 700 to the baby's crying. We retrieve the information provided by the detector using basic arduino function:

```
analog.Read(Sound_Pin);
```

2.7 Piezo buzzer

In our device we decided to use Piezo buzzer. It is based on an inverse principle of Piezoelectricity. It is the phenomena of generating electricity when mechanical pressure is applied to certain materials and the vice versa is also true. Such materials are called piezoelectric materials. Piezoelectric materials are either naturally available or manmade. Piezoceramic is class of manmade material, which poses piezoelectric effect and is widely used to make disc, the heart of piezo buzzer. When a small DC voltage is applied to the input pins, it is first converted to an oscillating signal using the combination of resistor and transistor. These oscillating signals are amplified using the inductor coil. When high voltage alternating signals are applied to the piezo ceramic disc, it causes mechanical expansion and contraction in radial direction. This causes the metal plate to bend in opposite direction. When metal plate bends and shrinks in opposite direction continuously it produces sound waves in the air.
Buzzer uses basic arduino function to produce a square wave of a specified frequency:

```
tone(pin, frequency, duration);
```

In our code those values are: 1, 440(Hz), 100 (ms) respectively.

2.8 RTC - DS1307

Our Real Time Clock uses I2C bus, since it is able to emit data as well as receive it. We decided to use popular RTClib.h library. In order to enable it to use I2C interface we need to connect SDA and SCL pins of our Arduino and the device itself. After connecting those pins, DS1307 clock does not require any further configuration. All that needs to be done is creating a RTC_DS1307 object:

```
#include <RTClib.h>
RTC_DS1307 rtc;
```

We use a functionality from the abovementioned library, which enables us to synchronise the RTC with the computer's clock during uploading the code to the device:

```
rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
```

Later we use the library's function to retrieve form the number of seconds passed since 1.01.1970 the actual time:

```
DateTime t = rtc.now();
```

Then to get particular time values we just use for instance:

```
t.hour();
t.minute();
t.second();
```

2.9 Wireless Radio Module - nRF24L01

In our project we use two Wireless Radio Modules, model nRF24L01, since it makes it easy to send data from one device to other. This module works in 2.4GHz frequency band and uses SPI interface. In order to implement its functionality we decided to use RF24.h library. After connecting SCK, MISO, MOSI and SS pins required for SPI interface we needed to create an object where we define Chip Enable and Ship Select Not pins respectively.

.

```
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(3, 2); // CE, CSN
```

Then in case of emitter we need to initiate the connection, open writing pipe and stop listening, essentially making emitter a Master. After that we are able to use function `write()` to emit desired information.

```
radio.begin();
radio.openWritingPipe(address);
radio.stopListening();

radio.write(&message, sizeof(message));
```

Receiving process looks a bit different. After initiation of the connection it opens reading pipe and starts listening. After that, in a loop, it checks whether a buffer is full and if so it reads the message.

```
radio.begin();
radio.openReadingPipe(0, address);
radio.startListening();

 if (radio.available()) {
        radio.read(&frame, sizeof(frame));
}
```

3. Failure Mode and Effect Analysis

Failure of certain modules is generally harmless and causes only one functionality to stop working, yet failure of others may end up making device unusable. Table below presents list of modules/functionalities failures with probability of their occurrence, gravity of such failure and description of detectability and device's reaction.

Risk	Probability	Gravity	Detectability	Reaction
Microcontroller failure	low	critical	Straightforward, after any interaction with the device	May cause any weird behavior of the the devices such as screen stuttering or random sign displaying. It may also cause any module to stop working properly or even stop working at all.
Power supply	low	critical	Straightforward, after any interaction with the device	May switch off the devices making it unusable or even brake the device. The worst scenario may happen after connecting battery with voltage not between 9 and 12 volt.
Wireless module failure	medium	critical	Straightforward, temperature and humidity displays random values and alarms do not ring	Causes total lack of communication between devices. Parent device is no longer is able to properly display humidity or temperature. In addition receiver device is not able to detect weather baby cries or not. Adequate alert is being displayed then.
RTC clock failure	low	medium	After entering in adequate menu option	May cause not precise time and date displaying. Also pre-set alarms may ring at wrong time or not at all.
Buzzer failure	low	low	Only after alarm activation	Limits feedback from alarm to just visual alert on display. Functions such as baby cry or pre-set alarms will still work and will give voice feedback will

LCD KeyPad failure	low	critical	Straightforward, LCD is either off or blue-lit and buttons do not work properly.	Prevents from any input to the device and limits all visual output.
Humidity/ temperature sensor failure	low	medium	Only after choosing temperature or humidity section in the menu.	Displays random values in temperature and humidity sections or last proper values received.
Sound sensor failure	low	critical	Straightforward, visible in parent's device	Failure of this module causes program to constantly send cry alarm to the parent's device making device unusable.

4. References

- Arduino Uno R3 - (https://www.fecegypt.com/uploads/dataSheet/1522237550_arduino%20uno%20r3.pdf)
- ATmega328P - (http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega328PB-Automotive-Data-Sheet-40001980B.pdf)
- Wireless Module - nRF24L01 - (https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf)
- RTC - DS1307 - (https://www.sparkfun.com/datasheets/Components/DS1307.pdf)
- Temperature and Humidity Sensor - DHT11 - (https://www.amouser.com/ds/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf)
- Sound Sensor Module - VMA309 - (https://www.robotshop.com/media/files/pdf/sound-sensor-module-arduino-datasheet.pdf)
- LCD KeyPad Shield - (https://www.robotshop.com/media/files/pdf/wiki-dfr0009.pdf)

5. Other information

In order for the wireless communication to work, emitting and receiving devices have to be within 2 meters range, because of the low power consumption set on both wireless modules and no antennas connected.