1. Laboratory stand

The laboratory stand consists of a PC, Intel evaluation board Galileo Gen2, a set of wiring and basic sensors cooperating with the Base Shield from the Grove Starter Kit Plus Intel IoT Edition (Fig. 1).

Grove Starter Kit Plus for IoT Intel Edition includes also:
- transition plate (Base Shield) to enable connection sensors to the Intel Galileo Board,
- alphanumeric LCD display (2 lines of 16 characters) with backlight RGB,
- push button,
- rotary potentiometer,
• electromagnetic relay module,
• buzzer module,
• microphone module,
• light sensor,
• temperature sensor,
• touch sensor,
• servo,
• three LED diodes (blue, green and red) cooperating with LED module Socket Kit,
• battery adapter,
• information brochure,
• 8GB SD card.

2. Programing in Arduino IDE

Turn on PC computer indicated by the teacher. At the time of its boot you can refer to the construction of the Galileo Gen 2 Intel board. Locate power supply socket and the USB Client port. Gently slide the Grove Base Shields into the slot on Intel Galileo Gen Board.

Fig. 2. Device Manager in Windows 8.1.
Connect the power supply to the Intel Galileo Gen2 board, and after that connect the USB cable to the computer using the USB client port (micro USB).

**Note 1!**
Keep the connection sequence of the power supply and USB cable. Reversing of the order wiring connection can damage Intel Galileo Board.

In the Device Manager in the *Porty* field (COM & LPT) check the COM port number at which the Intel Galileo Gen2 is visible. (Fig. 2).

Find a shortcut to the Arduino IDE, which should be on the Windows desktop and start the program. On the *Narzędzia* menu, scroll to the Intel platform Galileo Gen2 as cooperating with the Arduino IDE environment as shown in Figure 3.

![Fig. 3. Structure of menu *Narzędzia* of the Arduino IDE](image)

Then, using the menu *Narzędzia* -> *Port szeregowy*, select the right COM port, to which the motherboard of Intel Galileo Gen2 - read earlier in the Windows Device Manager is connected.

From the menu *Plik* -> *Przykłady* -> *01.Basics* (Fig. 4) load the Blink program. This is the simplest example of a program whose goal is blinking LED diode, located on the Galileo Intel Gen2 board on the left side from the USB HOST port. This LED diode is marked with symbol "L".

Compiling the programs (the so-called Sketch-s) in the Arduino IDE and their upload to the target board follow after pressing the left arrow icon (Fig. 4).

Examine the code and run the sample program (sketch) Blink and observe its operation. Modify the program by changing ON and OFF time of the LED diode "L" and changing sequences of flashing.

Intel Galileo Gen2 has two ports of UART type. The first (UART0) is derived for 2-pin connector pin (female) compatible with the Arduino platform (pin 0-RX, Pin 1-TX). The second port (UART1) is derived for 6-pin connector pin (male – goldpin type) located next to the Ethernet port. It is used mainly for communication with Linux Yocto, but it can also be used in the Arduino IDE. To use of this interface, it is necessary to use the USB compatible FTDI (or equivalent), wherein the signal converter UART TTL 3.3V for USB is integrated (Fig. 5).
Fig. 4. Sample program (sketch) in Arduino IDE

```c
/*
 Blink

 Turns on an LED on for one second, then off for one second, repeatedly.

 This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup(){
    // initialize the digital pin as an output.
    pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop(){
    digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000); // wait for a second
    digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
    delay(1000); // wait for a second
}
```

Fig. 5. The way to connect FTDI cable (UART TTL 3,3V->USB) to Intel Galileo Gen2 platform
Connect the FTDI cable as shown in Figure 5, making sure that the black wire is connected with terminal GND pin connector. Plug the USB cable to the PC port. Check in the Device Manager of Windows COM port number from which UART1 interface of Intel Galileo Gen2 platform will be seen.

Start the terminal program Tera Term, and then from menu File select New connection option. In the window that appears, select Serial and select the COM port previously read in the Device Manager (Fig.6). Then, from the Setup menu, select Serial port and set the baud rate to 115200 baud as shown in Figure 7. In this way, Tera Term terminal program is configured to work with Intel Galileo Gen2.

---

**Note 2!**

In sketches written in the Arduino IDE serial port UART0 it is mapped to a Serial port, and serial port UART1 to a SERIAL2 port. Port serial1 is not available!
In the next part of the lab analyze sample programs provided by the producer of Grove Starter Kit Plus Intel IoT Edition. These programs can be found in the menu *Plik -> Przykłady -> Grove Starter Kit V2*. When analyzing individual programs use the elements of the starter kit, which connect according to the booklet that came with the kit, using the supplied cables. Modify in accordance with Note 2 examples of programs that use UART interface (Serial Arduino IDE) and also make sure that the baud rate in programs Arduino IDE and in the terminal program Tera Term is the same.

**Note 3!**

Before connecting more elements to the transition board (Base shield) it is recommended to turn the power off of Intel Galileo board, and before unplugging the USB cable CLIENT (micro USB Connector).

Fig.8. The block diagram of the Intel Galileo Gen2.

In the protocol note the functions syntaxes, which will be useful later in laboratory:

a) read the status of the buttons,

b) set the status on the digital ports,

c) read values from analog-to-digital converter,
d) configuration of PWM channels,

e) displaying text on the LCD display and setting backlight RGB color.

3. Programming in Bash (Linux Yocto)

Controlling Galileo Intel platform states of ports in Linux is possible through the use of virtual file system sysfs. It should be noted, that due to the difference in the construction of the Galileo Gen1 platform (using chip Cypress CY8C9540A) and Galileo Gen2 (using chip PCAL9555A) programming both platforms at the level of Linux is slightly in different way. This study is limited to discussion of second generation Galileo platform (Fig. 8).

Connect the FTDI as shown in Figure 5 and run Tera Term program, and then configure it to connection via the serial port with a speed of 115,200 bps (Fig. 6 and 7). Connect the power of the Intel Galileo Gen2 board. Wait to load a Linux Yocto located in Flash memory. Log in to the operating system as: root.

GPIO ports status of the Galileo Intel platform running under Linux can be checked in the file "/sys/kernel/debug/gpio". To read its contents issue the command in terminal (in the program window Tera Term):

```
less /sys/kernel/debug/gpio
```

![Fig. 9. Contents of the file GPIO](image)

The effect of the command will be listing of the file’s contents as shown in Figure 9. It should be noted that the Intel Galileo Gen2 platform has 79 ports, but not all are routed to connectors compatible with Arduino. Most of them are used to configure the ports compatible with Arduino as shown in Table 1. Some ports are native and controlled directly from Intel Quark, others are...
connected by intermediate device such as PCAL9555 and PCA9685 (block GPIOexp, PWM, MUX, SHIFT on figure 9).

Table 1. Configuration of Arduino-compatible ports on Galileo Gen2 platform

<table>
<thead>
<tr>
<th>Shield pin</th>
<th>Function</th>
<th>Linux</th>
<th>Level Shifter</th>
<th>GPIO Exp</th>
<th>22k Pull-Up</th>
<th>Pin Mux 1</th>
<th>Pin Mux 2</th>
<th>Interrupt modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>I00</td>
<td>UART0 RX</td>
<td>ttyS0</td>
<td>gpi032</td>
<td>gpi033</td>
<td></td>
<td></td>
<td></td>
<td>L/H/R/R/F</td>
</tr>
<tr>
<td></td>
<td>GPIO1</td>
<td>gpi011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I01</td>
<td>UART0 TX</td>
<td>ttyS0</td>
<td>gpi028</td>
<td>gpi029</td>
<td>gpi045 (H)</td>
<td></td>
<td></td>
<td>L/H/R/R/F</td>
</tr>
<tr>
<td></td>
<td>GPIO2</td>
<td>gpi012</td>
<td></td>
<td></td>
<td>gpi045 (L)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I02</td>
<td>UART1 RX</td>
<td>ttyS1</td>
<td>gpi034</td>
<td>gpi035</td>
<td>gpi077 (L)</td>
<td></td>
<td></td>
<td>L/H/R/R/F</td>
</tr>
<tr>
<td></td>
<td>GPIO3</td>
<td>gpi013</td>
<td></td>
<td></td>
<td>gpi077 (L)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I03</td>
<td>UART1 TX</td>
<td>ttyS1</td>
<td>gpi014</td>
<td>gpi017</td>
<td>gpi076 (H)</td>
<td>gpi064 (L)</td>
<td></td>
<td>L/H/R/R/F</td>
</tr>
<tr>
<td></td>
<td>GPIO4</td>
<td>gpi015</td>
<td></td>
<td></td>
<td>gpi076 (L)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PWM1</td>
<td>pwm1</td>
<td></td>
<td></td>
<td>gpi076 (L)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I04</td>
<td>GPIO5</td>
<td>gpi062</td>
<td></td>
<td></td>
<td>gpi076 (L)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I05</td>
<td>GPIO6</td>
<td>gpi06</td>
<td>gpi036</td>
<td>gpi037</td>
<td></td>
<td></td>
<td></td>
<td>R/F/B</td>
</tr>
<tr>
<td>I06</td>
<td>GPIO7</td>
<td>gpi018</td>
<td>gpi019</td>
<td>gpi066 (L)</td>
<td></td>
<td></td>
<td></td>
<td>R/F/B</td>
</tr>
<tr>
<td>I07</td>
<td>GPIO8</td>
<td>gpi064</td>
<td>gpi065</td>
<td>gpi066 (L)</td>
<td></td>
<td></td>
<td></td>
<td>R/F/B</td>
</tr>
<tr>
<td>I08</td>
<td>GPIO9</td>
<td>gpi010</td>
<td>gpi020</td>
<td>gpi021</td>
<td>gpi068 (L)</td>
<td></td>
<td></td>
<td>R/F/B</td>
</tr>
<tr>
<td>I09</td>
<td>PWM2</td>
<td>pwm7</td>
<td>gpi023</td>
<td>gpi024</td>
<td>gpi068 (L)</td>
<td></td>
<td></td>
<td>R/F/B</td>
</tr>
<tr>
<td>I10</td>
<td>GPIO10</td>
<td>gpi016</td>
<td>gpi026</td>
<td>gpi027</td>
<td>gpi074 (L)</td>
<td></td>
<td></td>
<td>L/H/R/R/F</td>
</tr>
<tr>
<td>I11</td>
<td>GPIO11</td>
<td>gpi017</td>
<td>gpi028</td>
<td>gpi029</td>
<td>gpi074 (L)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPI MISO1</td>
<td>gpi05</td>
<td>gpi030</td>
<td>gpi031</td>
<td>gpi072 (L)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPI MISO0</td>
<td>gpi06</td>
<td>gpi032</td>
<td>gpi033</td>
<td>gpi072 (L)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I12</td>
<td>SPI MCK1</td>
<td>gpi07</td>
<td>gpi034</td>
<td>gpi035</td>
<td>gpi072 (L)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I13</td>
<td>SPI MCK0</td>
<td>gpi08</td>
<td>gpi036</td>
<td>gpi037</td>
<td>gpi072 (L)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I14</td>
<td>ADC A0</td>
<td>in_voltage0_raw</td>
<td>gpi049</td>
<td>gpi049</td>
<td></td>
<td></td>
<td></td>
<td>R/F/B</td>
</tr>
<tr>
<td>I15</td>
<td>GPIO10</td>
<td>gpi050</td>
<td>gpi051</td>
<td>gpi052</td>
<td></td>
<td></td>
<td></td>
<td>R/F/B</td>
</tr>
<tr>
<td>I16</td>
<td>ADC A1</td>
<td>in_voltage1_raw</td>
<td>gpi053</td>
<td>gpi053</td>
<td></td>
<td></td>
<td></td>
<td>R/F/B</td>
</tr>
<tr>
<td>I17</td>
<td>GPIO11</td>
<td>gpi054</td>
<td>gpi055</td>
<td>gpi056</td>
<td></td>
<td></td>
<td></td>
<td>R/F/B</td>
</tr>
<tr>
<td>I18</td>
<td>ADC A2</td>
<td>in_voltage2_raw</td>
<td>gpi057</td>
<td>gpi057</td>
<td></td>
<td></td>
<td></td>
<td>R/F/B</td>
</tr>
<tr>
<td>I19</td>
<td>GPIO12</td>
<td>gpi058</td>
<td>gpi059</td>
<td>gpi060</td>
<td></td>
<td></td>
<td></td>
<td>R/F/B</td>
</tr>
<tr>
<td></td>
<td>ADC A3</td>
<td>in_voltage3_raw</td>
<td>gpi061</td>
<td>gpi061</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There are the following designations in the table:
- "L" – GPIO port is configured as output in low state
- "H" – GPIO port is configured as output in high state
- "I" – GPIO port is configured as input in high impedance state

**Write digital ports**

The sample configuration IO13 port that is connected to the LED "L" on the Galileo Gen2 board is as follows:

According to Table 1 IO13 port is connected to the gpio7 Intel Quark port, so we export gpio7 port to be able to be controlled with a virtual file system sysfs:
Next we set gpio7 port as output port:

```
echo -n "7" > /sys/class/gpio/export
```

Sequentially we set high state on gpio7 port to turn on the LED diode "L":

```
echo -n "out" > /sys/class/gpio/gpio7/direction
```

```
echo -n "1" > /sys/class/gpio/gpio7/value
```

Is LED diode "L" turn on? If not, it is still necessary to configure multiplexers (Fig. 8), which mediate between the microprocessor ports and connectors compatible with Arduino. To do this, according to Table 1, the processor gpio30 ports and gpio46 should be configured as output in the low state. While using the IO13 port as output configuration resistors of pullup and pulldown types (using gpio31) can be omitted. Therefore, issue the commands:

```
echo -n "30" > /sys/class/gpio/export
```

```
echo -n "46" > /sys/class/gpio/export
```

```
echo -n "out" > /sys/class/gpio/gpio30/direction
```

```
echo -n "out" > /sys/class/gpio/gpio46/direction
```

```
echo -n "0" > /sys/class/gpio/gpio30/value
```

```
echo -n "0" > /sys/class/gpio/gpio46/value
```

The above command should result in the illumination of the LED "L" placed next to the USB HOST port and connected to the port IO13 compatible with Arduino. The state of this port can now be changed by changing only the status on Intel Quark gpio7 port.

```
echo -n "0" > /sys/class/gpio/gpio7/value
```

```
echo -n "1" > /sys/class/gpio/gpio7/value
```

With the command:

```
less /sys/kernel/debug/gpio
```

check states of gpio ports after configuration of IO13 port after above commands are executed, compare them with Figure 9, the differences write in protocol performed exercise.

**Read digital ports**

Each port IO compatible with Arduino can be programmed as input. For example for IO2 port issue the commands:

```
echo -n "13" > /sys/class/gpio/export
```

```
echo -n "34" > /sys/class/gpio/export
```

```
echo -n "35" > /sys/class/gpio/export
```

```
echo -n "77" > /sys/class/gpio/export
```

```
echo -n "in" > /sys/class/gpio/gpio13/direction
```

```
echo -n "out" > /sys/class/gpio/gpio34/direction
```

```
echo -n "out" > /sys/class/gpio/gpio35/direction
```

```
echo -n "out" > /sys/class/gpio/gpio77/direction
```
State of IO2 port can be read with command:

```
cat /sys/class/gpio/gpio13/value
```

In order to better illustrate the changes in the IO2 port connect to the Base Shield from the Grove starter kit Button switch or Touch to the D2 port and issue the following command in Terminal (in the Tera Term program).

```
while [ 1 ]
do
cat /sys/class/gpio/gpio13/value
done
```

By changing the button state observe the values returned by Linux. In the protocol answer the questions. Does the switch connect port IO2 to ground or to a power line?? In such a situation do we use pullup or pulldown resistors?

**Note 4!**
Loops can be interrupted by pressing Ctrl + C.

**Reading the states of analog inputs (ADCs)**

Analog inputs of Intel Galileo platform utilize analog-to-digital converter AD7298. This is an 8-channel transmitter, but only 6 channels connected to the ports A0-A5 Arduino-compatible are used. Used ADC has a resolution of 12 bits, therefore returns a value in the range of 0-4095.

From Linux reading an analog input (eg. A0) is implemented, as well as digital inputs, using the sysfs virtual file system by using the command:

```
cat /sys/bus/iio/devices/iio:device0/in_voltage0_raw
```

However, earlier properly configured connections of analog port compatible with Arduino standard with the input analog-to-digital by appropriate multiplexers should be checked. As indicated in Table 1 ports A0-A3 are by default configured as analog inputs. Ports A4 and A5 require the configuration like as in the case of digital inputs and outputs.

Connect the potentiometer from set of Grove Starter Kit Plus Intel IoT Edition to A0 port. Run the following script:
By changing the position of the potentiometer observe the data returned by the analog-to-digital converter. Save in the protocol extreme values from the transmitter. Disconnect potentiometer and observe and write down what the values will return the ADC converter. For analog port A0 using the Intel Quark port gpio49 connect pullup resistors and next pulldown and observe how they impact on the performance of the transmitter in case the connected and disconnected potentiometer. Make a note in protocol.

**Programming of PWM (Pulse-Width Modulation) channel**

The PWM signal fulfills a variety of functions in embedded systems. Mostly it is applied to control the brightness of the LEDs, the rotation speed of motors or servomechanism control. Galileo Gen2 board has a 16-channel 12-bit PWM controller (PCA9685) that is connected to the X1000 Quark processor through the I²C interface. From the 16 channels, which belong to the PCA9685 device only 6 channels are used as PWM outputs that are wired to the connector compatible with the Arduino (Fig.8). The others are used as output GPIO ports (General Purpose Input/Output).

Period of PWM signal in Intel Galileo Gen2 platform is configured simultaneously for all channels and can be in the range of 666,666 to 41,666,666 nanoseconds.

In the protocol calculate the maximum and minimum frequency of the PWM waveform, which can generate Galileo Gen2 platform.

Access to the configuration of PWM channels is available via a file system sysfs and exactly via the directory: /sys/class/PWM/pwmchip0/.

Just as for ordinary GPIO ports to gain access to the channel PWM (eg. Channel 1) it should be exported using the command:

```bash
echo -n "1" > /sys/class/pwm/pwmchip0/export
```

Then running the channel of first PWM module follows by issuing the command:

```bash
echo -n "1" > /sys/class/pwm/pwmchip0/pwm1-enable
```

The period of the generated waveform (eg. 20 ms) is given in nanoseconds with the command:

```bash
echo -n "20000000" > /sys/class/pwm/pwmchip0/device/pwm_period
```

Next, enter the duty cycle, specified in nanoseconds calculated according to the formula:

\[
duty \_ cycle[\text{ns}] = pwm \_ period[\text{ns}] \cdot \frac{k_w [\%]}{100}\%
\]  
(1)

where: \(k_w\) - duty cycle expressed as a percentage.
For example, for the waveform with duty cycle equals 50% and frequency 50 Hz the command setting the duty cycle takes the form:

```bash
echo -n "10000000" > /sys/class/pwm/pwmchip0/pwm1/duty_cycle
```

Of course, it is still necessary to configure multiplexers (Fig 8), which mediate between the microprocessor ports and connectors compatible with Arduino. Thus, as shown in Table 1, the entire setup procedure of the first channel PWM so as to generate a waveform with a frequency of 50 Hz with the duty cycle of 50% is as follows:

```bash
  echo -n "16" > /sys/class/gpio/gpio16/export
  echo -n "17" > /sys/class/gpio/gpio17/export
  echo -n "76" > /sys/class/gpio/gpio76/export
  echo -n "64" > /sys/class/gpio/gpio64/export
  echo -n "out" > /sys/class/gpio/gpio16/direction
  echo -n "in" > /sys/class/gpio/gpio17/direction
  echo -n "out" > /sys/class/gpio/gpio76/direction
  echo -n "out" > /sys/class/gpio/gpio64/direction
  echo -n "0" > /sys/class/gpio/gpio16/value
  echo -n "0" > /sys/class/gpio/gpio76/value
  echo -n "1" > /sys/class/gpio/gpio64/value
  echo -n "1" > /sys/class/pwm/pwmchip0/export
  echo -n "1" > /sys/class/pwm/pwmchip0/pwm1/enable
  echo -n "20000000" > /sys/class/pwm/pwmchip0/device/pwm_period
  echo -n "10000000" > /sys/class/pwm/pwmchip0/pwm1/duty_cycle
```

In order to validate the generated waveform connect blue LED to the IntelGalileo Gen2 platform via the D3 connector of the Base Shield from Grove Starter Kit. By varying the duty ratio in 0÷100% range watch the lighting of the LED.

### 4. Final task:

1. Write a program in the Arduino IDE environment, which will implement the function of the heating controller with hysteresis ± 0.5 °C. Reading the current temperature will be implemented using a thermistor with a Grove starter kit. Setting the temperature by 1 °C will be implemented using the momentary Button (T+) and Touch sensor (T-), switching on the heater will be done by the relay module. On the LCD screen will be presented set and measured temperature. Dependences on the measured and set temperature should be visualized as the color of the backlight according to the algorithm: If the actual temperature is lower than the set - red (heater ON), in the reverse situation - blue (heater OFF). If both temperatures are the same – green color.
2. Write Bash script using the Vi editor enabling servomechanism control (port IO5) using a potentiometer connected to the IntelGalileo Gen2 platform (port A1). Take into account the fact that the servomechanism accepts a PWM signal 50 Hz and duty cycle in the range of 4÷12%. To help you can take advantage of proposed or other available online courses of Bash and the vi editor (remember to set access rights for your own script with the command: `chmod + x script_name`, run the script: `./nazwa_skryptu`):

**VI:**

**Bash:**
- [http://www.uw-team.org/artykul_bash_kurs.html](http://www.uw-team.org/artykul_bash_kurs.html)