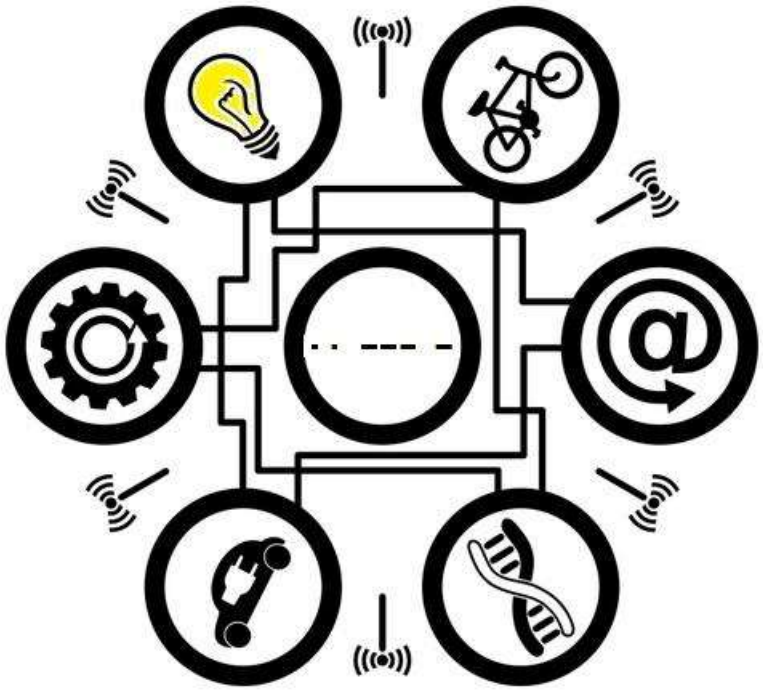
Workshop scenario



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

Workshop scenario ·· --- -

Aims:
- familiarising with the idea of the Internet of Things
- increasing curiosity of the participants and their creative thinking
- raising a positive attitude towards new technologies
- familiarising with electronic systems and basics of programming

Materials:
- educational film: https://www.youtube.com/watch?v=LlhmzVL5bm8
- pictures of sensors (annex 1) per group
- Morse alphabet (annex 2) per group
- components – 1 set per group of 2-3 people: ESP32 board, female-female cables, USB cable, Traffic Light LED Display Module for Arduino, relay
- a computer per group (with Internet access and arduino1.8.13. preinstalled)
- projector
- an account at *https://appinventor.mit.edu/* (Google account is needed)
- a previously prepared lamp:

Instruction of preparing a lamp:

To connect the 220V cable to the relay, first cut the cable in case it is made of one piece. You will notice that it is composed of 2 or 3 wires assembled together. Separe one wire from the group and cut it like in Figure 1.

Fig. 1. Cut a 220/230V cable.

Arduino1.8.13 installation:
https://www.youtube.com/watch?v=Y0r5dXnl5hQ

Class schedule:

1. **Introduction:**

Introductory film is available here: https://www.youtube.com/watch?v=LlhmzVL5bm8
You can choose your national language and make the subtitles large.

Discussion about the film. Ask the participants about their impression. What do they think about the film and the Internet of Things idea?

2. **Basics of electronics and programming:**

Basics of electronics. Work in groups.
Give an ESP32 board to each group and ask if the participants know what it is for. Explain that it is an electronic system that can send and receive information to/from different components. It is equipped with a WI-FI module so it can be connected not only to a computer but also to a local network or the Internet.
Distribute USB cables, female-female cables and LED diodes to every group. Explain that we will be wiring components together. The USB cable should be connected to a computer and an ESP32 board (Fig. 2). Make sure there are no other devices plugged to the other USB ports (e.g. mobile phones). The diodes can be connected to the board by female-female cables. Let the participants connect a chosen LED diode (a pin next to the diode) with any digital pin on the board (starts with 'D').

Fig. 2. Port for mini-USB cable on the ESP32 board.

Basics of programming. Work in groups.
Presentation of the TUNIOT tool. The tool is available on *easycoding.tn*. Put on a projector the *easycoding.tn* webpage so everyone can follow changes in the code. Select the desired language version of the TUNIOT FOR ESP32 on the left side toolbar (Fig. 3). Explain what a block programming is: from the menu on the left side you can choose pieces of a code which you can put in the middle panel. The main program is divided into two parts: 'Setup' which is executed only once and 'Main loop' – this part of the code will be repeated.

Fig. 3. Hyperlink to the TUNIOT tool on easycoding.tn

Exercise 1. Let's try to write our first program. It will turn on a LED diode. From the IN/**OUT** menu choose the Digital option because our diode is connected to a digital pin. Explain what the difference between digital and analog pin is. Digital pins receive and send zero-one signals like turn on – turn off. Only two states are possible: diode is ON or it is OFF. Some sensors can be connected to analog pins which give a broader range of information, e.g. we can measure temperature, humidity or luminosity in a range of values.

Then, choose **DigitalWrite PIN# D0 STAT HIGH** block and drag and drop it in the main program under the **Main loop** block (Fig. 4). The block should stick to the code with a characteristic sound. Check if the participants managed to do it. Inside that new piece of code we can change two things: pin number and its state. In the first case we choose the number of the pin where the diode is connected (e.g.: D19). The second part of the block should be set as 'HIGH'. 'HIGH' means that the diode is either ON, 'LOW' – it is OFF. The code for turning the diode on is ready now. Explain to the participants that the blocks are translated to a language that computers understand, in this case – C language. You can see how it looks by clicking the 'CODE' tab (Fig. 5). The first icon of the upper menu allows you to copy the code (Fig. 5, blue circle).

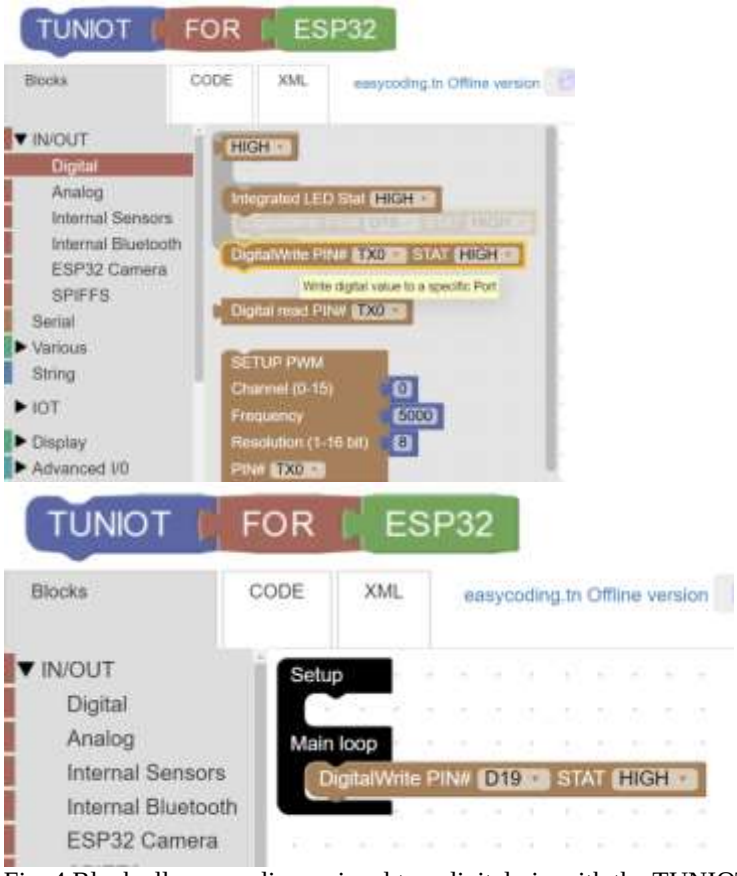


Fig. 4 Block allows sending a signal to a digital pin with the TUNIOT tool.

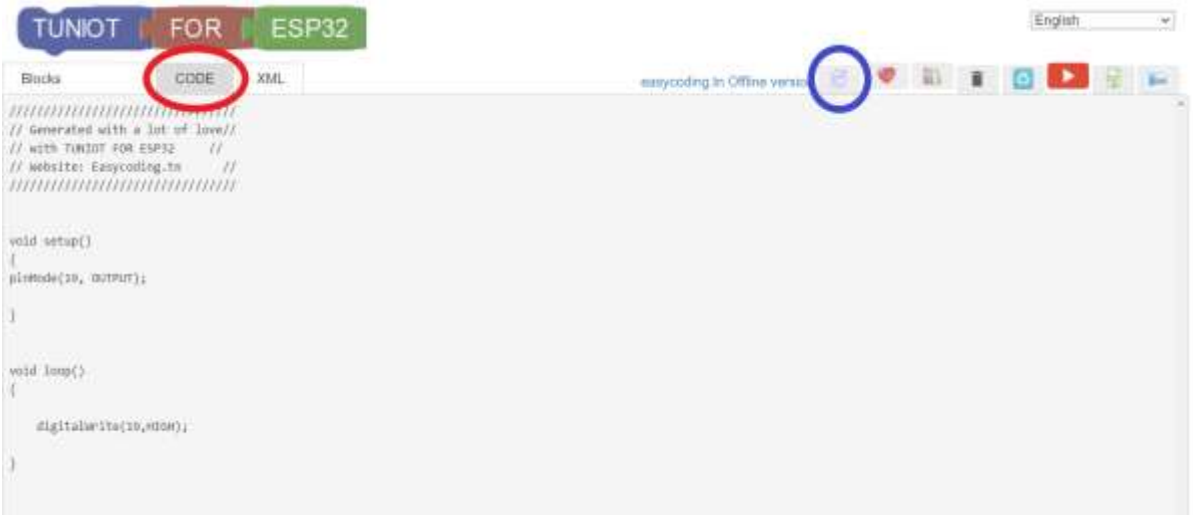


Fig. 5. A view of the TUNIOT tool with the C code and the copy option marked.

Paste the code to the arduino1.8.1 program. You can use Ctrl+A and Ctrl+V keybinds. Each program should be compiled and loaded to the ESP32 board. You can do it with the first two icons of the main menu (Fig. 6). When the loading is over, the diode should turn on. If that is not the case, check the wiring, the code and try to load once more.

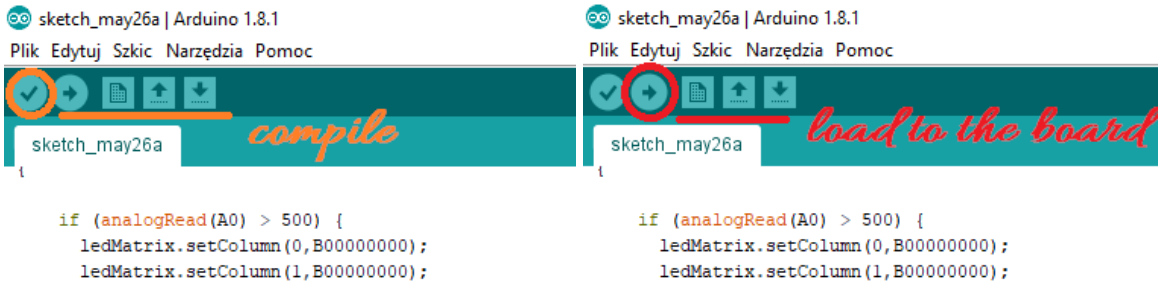


Fig. 6. Compiling and loading buttons in the arduino1.8.13 program.

Exercise 2. Make a diode blink (turning on and off by itself). Tell the participants to do the task unassisted. They will probably add the piece of code about turning off only. There is nothing wrong with the solution but you should explain that the time interval between turning on and off is so small (in the order of milliseconds) that a human eye cannot catch it. The diode seems to not be blinking at all. You need to add a delay after turning on and off pieces of code. You can find it in **Various -> Delay Ms**. The delay is set in milliseconds. The participants can try different values of the delay and check how the diode blinks. An exemplary code is presented in Fig. 7.

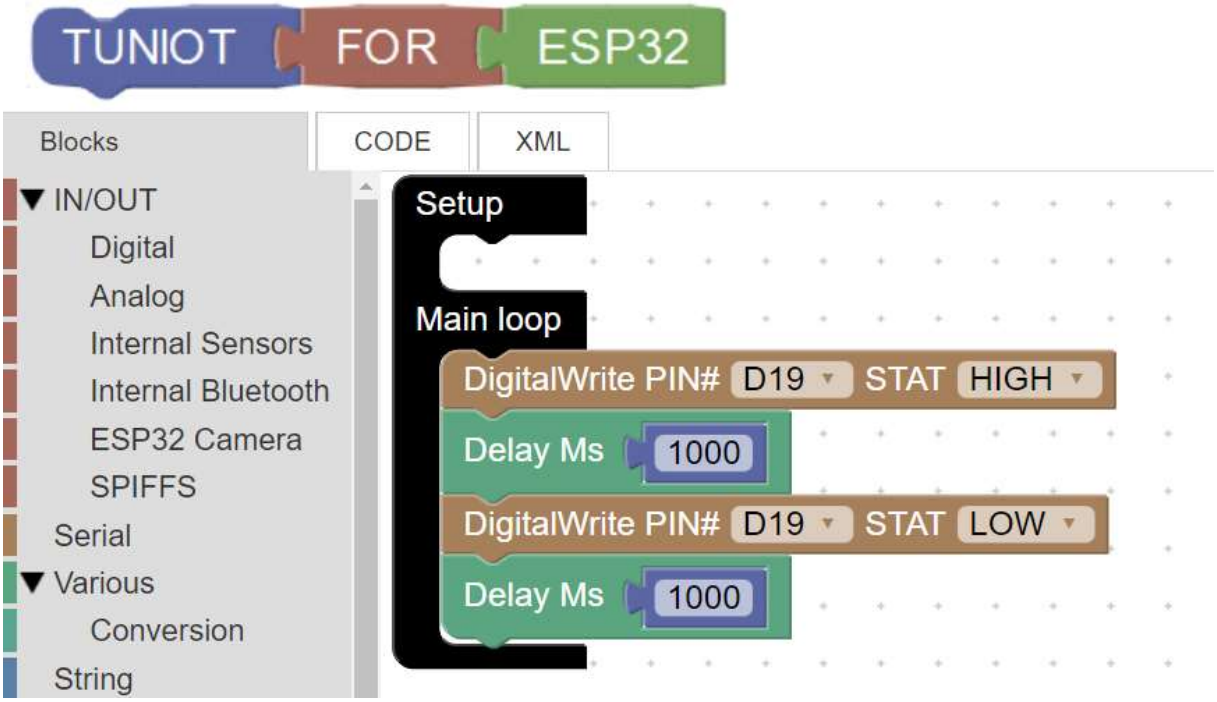


Fig. 7. Block code to the exercise 2.

Exercise 3. Repeat the exercise 2 but for 2 diodes (e.g. red and green) blinking synchronously (Fig. 8). Remember to correctly connect the next diode to a digital pin.

Fig. 8. Block code to the exercise 3.

Exercise 4. Modification of the exercise 3 – system of 2 diodes where one is blinking with a different speed than the other (Fig. 9).

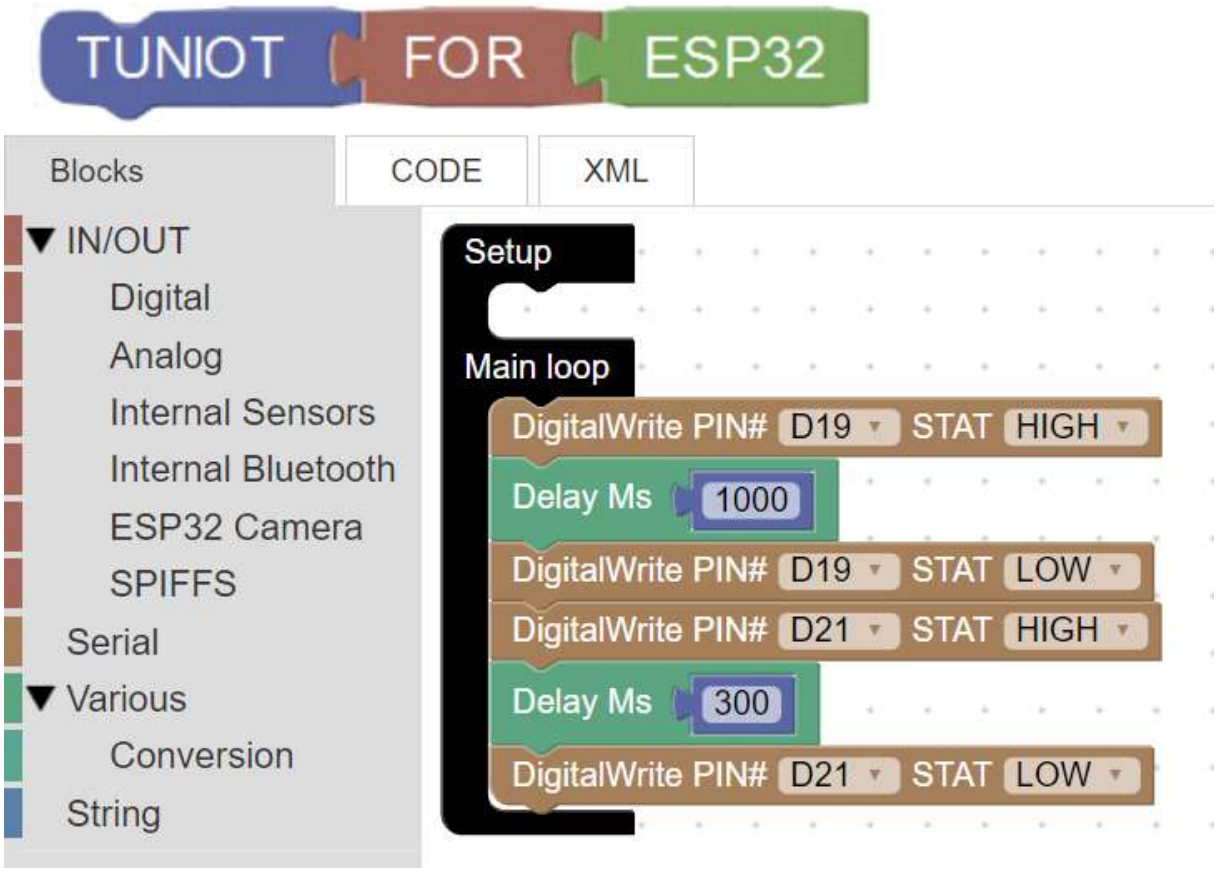


Fig. 9. Block code to the exercise 4.

Exercise 5. Turn the first diode on three times, then the second one three times and the third (Fig. 10). Explain the concept of loops. Code in the loop will be executed three times (from 1 to 3 with a step of 1). The loop block can be found in: **Loops -> count with**…

Exercise 6. Imitation of traffic lights. Tell the participants to try and make the task unassisted. Firstly, turn the green diode on for 5 seconds, then let the yellow diode blink for 3 seconds. In the end turn the red lamp on for 5 seconds (Fig. 11).

Fig. 10. Block code to the exercise 5.

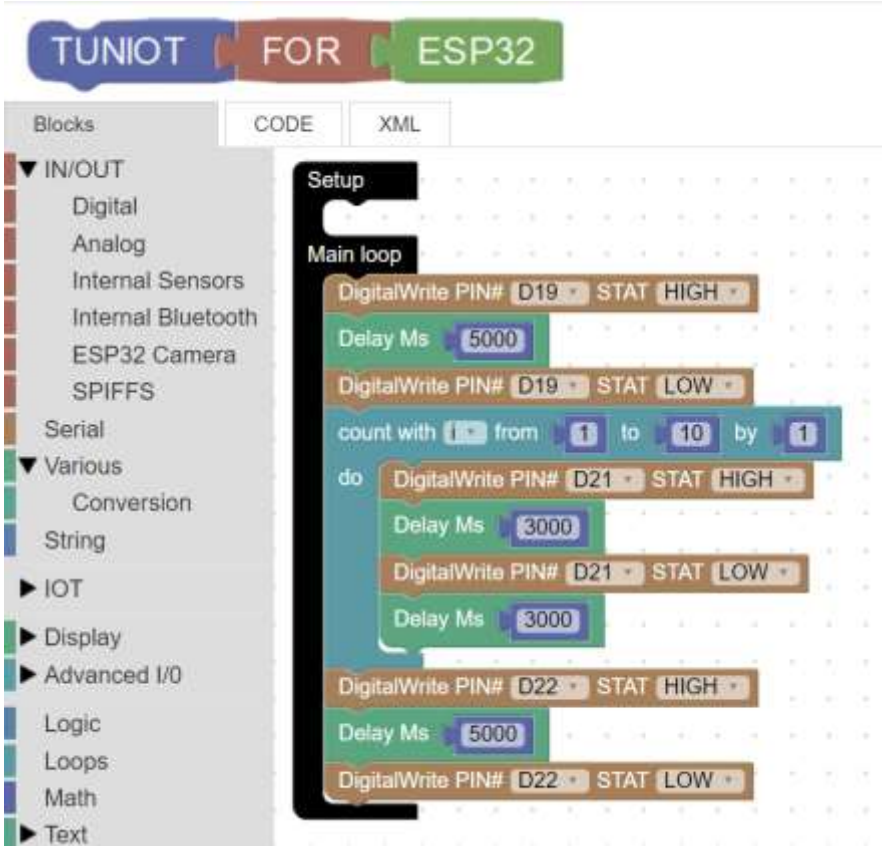


Fig. 11. Block code to the exercise 6

.Turning on/off a lamp. Presentation

CAUTION! This part of the workshop requires access to a 230V electric power supply. Do not let the students try it by themselves without any adult supervision.

Show the participants the relay component and explain it is a switcher that can switch off/on any electric device (e.g.: a kettle, lamp, microwave). Connect it to the ESP32 and the lamp as it is shown in Fig. 12. The resulting device should look like in the pictures below (Fig. 13).

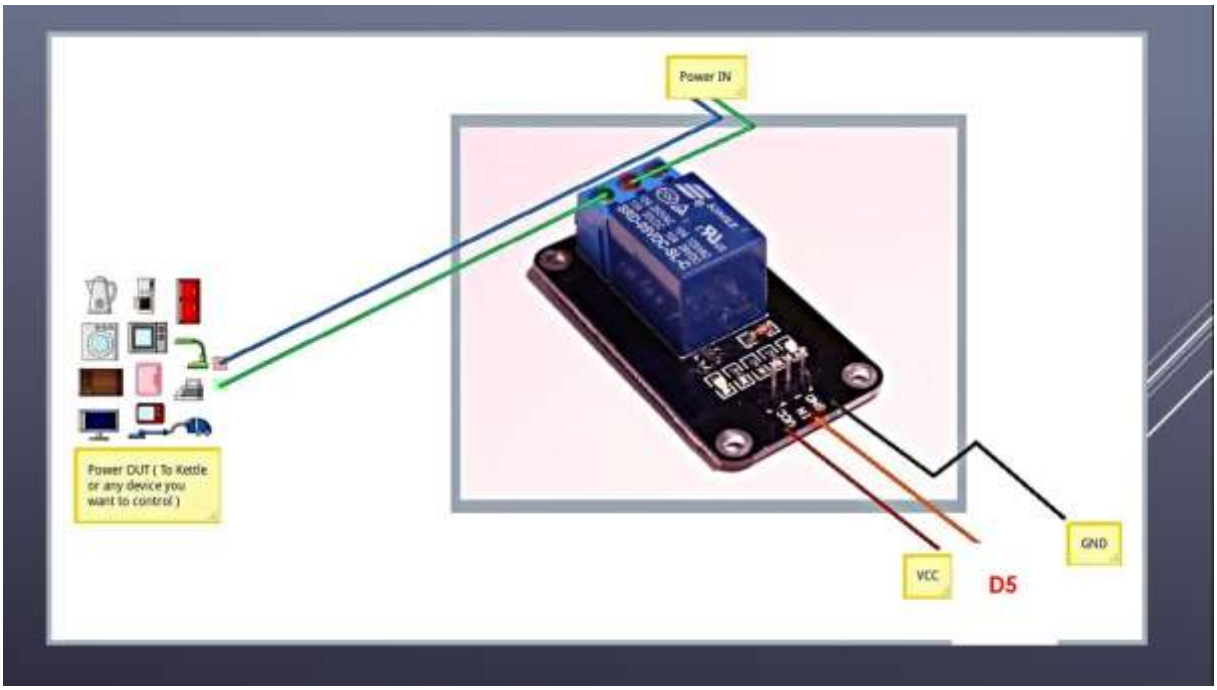


Fig. 12. How to wire a relay. Upper part (a lamp switch on/off cables). Lower part connect to your NODEMCU (VCC, any digital pin and GND pins)

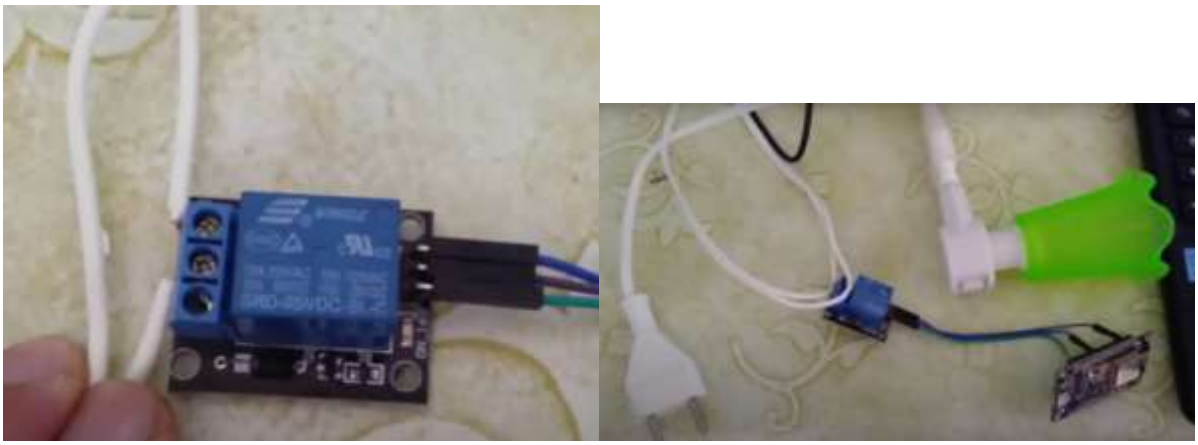


Fig. 13. Wiring a lamp to a relay.

Use the code from exercise 2 to blink the lamp. You can set a greater delay. Explain to the participants how they can control everyday devices using simple programs and the IoT concept. Ask them what they would control with a relay (only switch on/off devices).

Advanced part – controlling lamps by the Internet

If you have no time for the advanced part of the workshop, use the template code from the TUNIOT tool (Fig. 14) and go to Exercise 9.

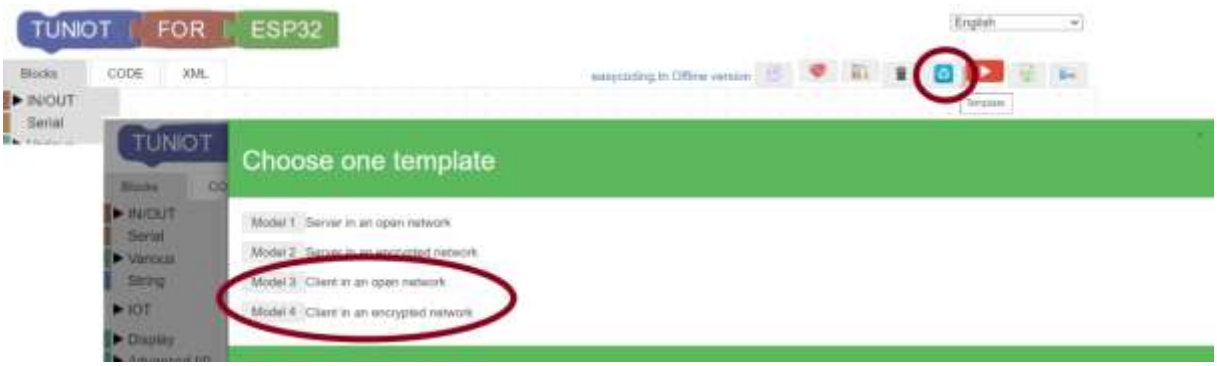
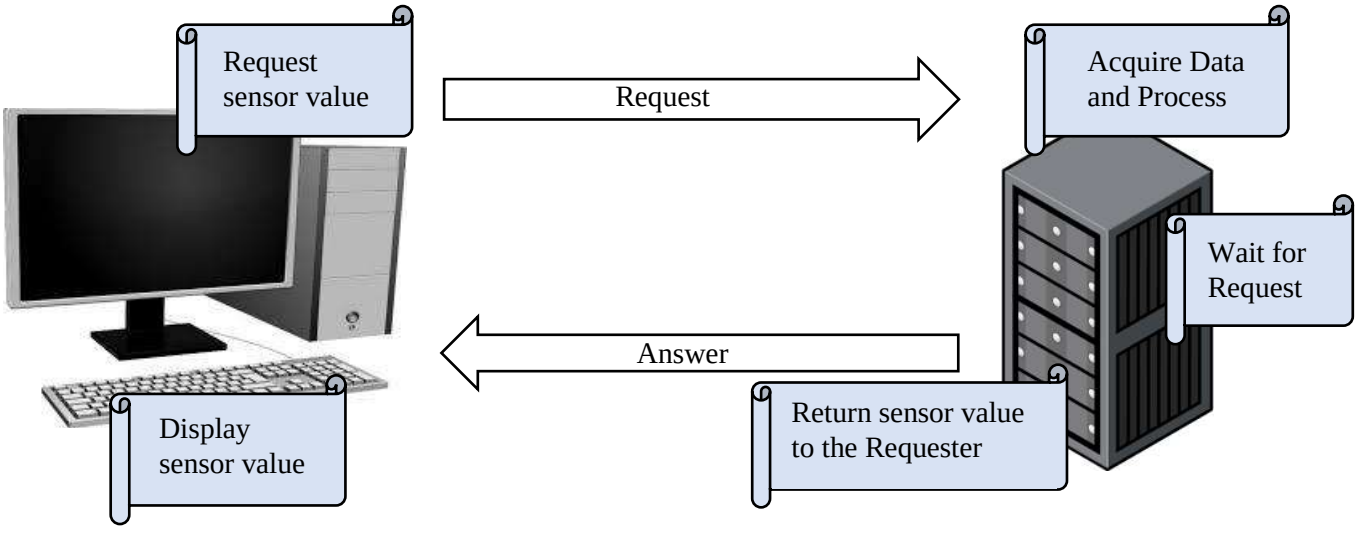


Fig. 14. Importing TUNIOT templates. Choose Model 3 if your network is not secured with a password or Model 4 if you need to enter the password to connect.

Client-server communication. Discussion.
Explain what the client-server communication is. You can use an analogy of a shop assistant and a buyer. The scheme below can be useful:



What is the client/ the server in our case? A computer or a NODEMCU board?
Give some examples of client-server architecture:
- e-mail: a program allowing to open / send messages is a client of an e-mail server. Mails are stored on the server
- WWW websites: a client sends a request to the WWW server from a browser. The server replies with a WWW webpage file (a HTML file) that can be decoded by the client.

Connecting to a local network. Work in groups.
Exercise 7. Set a hotspot from your computer or mobile phone. You can also use a local WI-FI if there is not a two-stage login process (e.g. with a browser). Drag the **Disconnect** block to the **Setup** in TUNIOT. It can be found here: **IOT -> IOT Station.** This command disconnects the board from any other networks that it could be connected to before. Set a short delay, ca. 3-5 seconds. This time is needed to open the Serial Monitor. Let's print „START" to check if the program works. Then, choose the **Connect Network SSID** block with or without a password from **IOT -> IOT Station**. You should give the network name in the first field and the password in the second (if the network is secured). The next step is to create a while loop where we will check if the connection succeeded. Explain how the while loop works: until the board is not connected with the local network, the program will print „ …". When the loop condition is satisfied, print a proper message (Fig. 14). Copy-paste the code to the arduino1.8.13 program and execute in a standard way. The messages are printed in the **Serial Monitor** which can be opened by clicking on the magnifier icon in the upper right corner (Fig. 15). Ask participants if they are successfully connected.

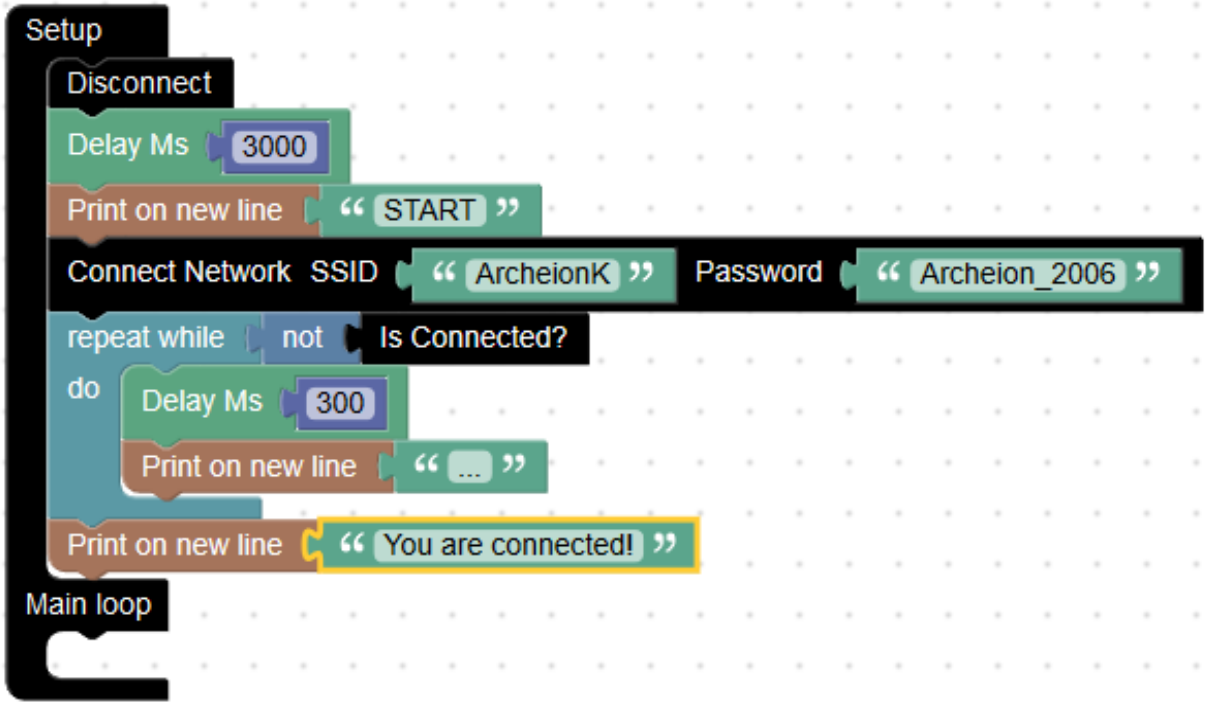


Fig. 14. Block code to the exercise 7.

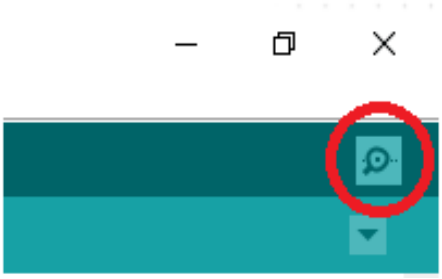


Fig. 15. The Serial Monitor icon in the arduino1.8.1 program.

Exercise 8. We will need the IP address of the NODEMCU board in order to connect it to the network. Explain that it is a parameter describing a device in a network. You need to add the **Print on new line** block with **Local IP** connected to it (Fig. 16). The IP address will be printed in the **Serial Monitor**.

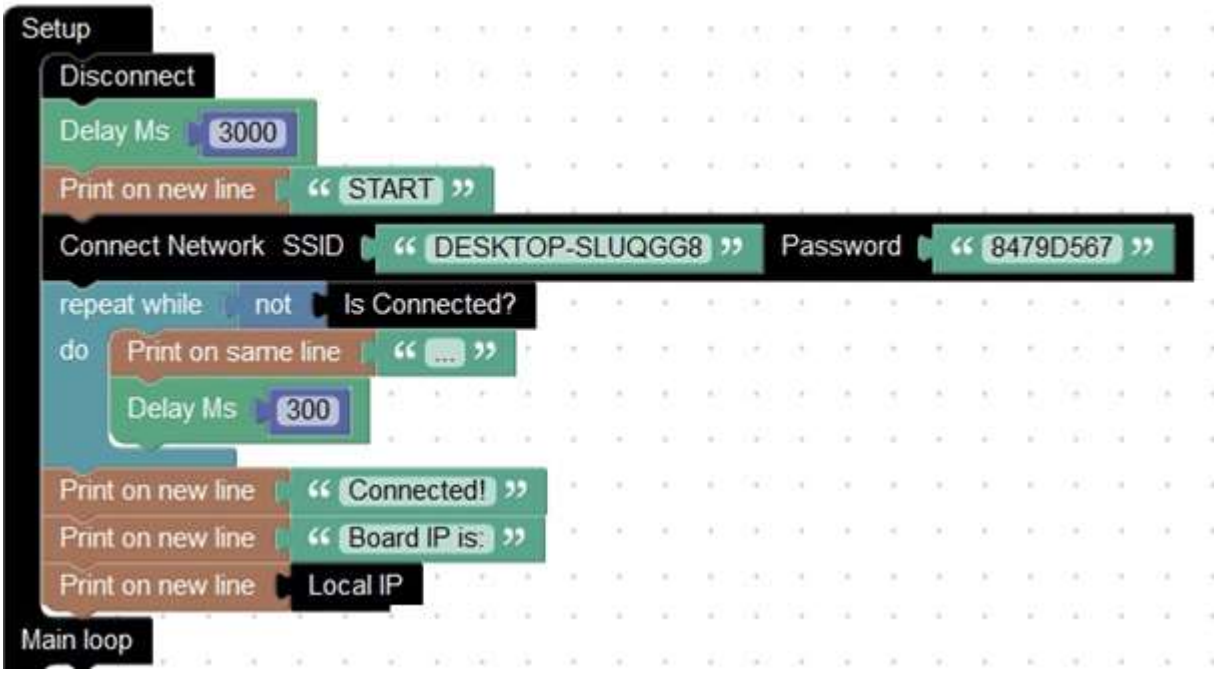


Fig. 16. Block code to the exercise 8.

Exercise 9. Now, we are going to settle the reaction of the board (server) for a request of turning on/off a diode/lamp. We need to declare a text ('string') value of the client message. Let's do it at the beginning of the code (in **Setup**) by adding **Declare >i< as String Value** from the String menu. We can change 'i' for another name e.g. ClientRequest. Add an empty text block from the **Text** menu. Now, we have an empty variable defined.
In the **Main loop** add a block **Wait connection** from **IOT -> IOT Server** menu. From the **String** menu add **Set STRING ClientRequest to** block below. The program automatically changes the name of a variable for the defined one. Come back to the **IOT -> IOT Server** menu and add the **Server Read request** block. Our variable is not empty now. It will be replaced with the request read by the server. In the String menu choose the second option from below: **ClientRequest clear HTTP request in Server**.
From the Logic menu choose the first block and add it below. Fill it with the following blocks. In the first line: **=** block (from **Logic**). Before the '=' sign add the **ClientRequest** block (from **String**), and after it - an empty text block from the **Text** menu. Change the text for 'ON'. In the second line (after 'do') add the **DigitalWrite PIN# D19 STAT HIGH** from the **IN/OUT -> Digital** menu. Copy the if loop and stick it below. This time we change 'ON' text for 'OFF' and 'HIGH' digital value for 'LOW'. This part of the code checks what the client request is. If it is 'ON' the board (server) will turn a diode on. If it is 'OFF' it will switch it off. The last thing is to add the **client flush** block from the **IOT -> IOT Server menu** (Fig. 17). It will clear the request so the program can read it again.
Execute the code and write down your local IP address. It will be printed in the **Serial Monitor** which can be opened by clicking on the magnifier icon in the upper right corner (Fig. 15) in the arduino1.8.13 program.

Fig. 17. Block code to the exercise 9.

3. Controlling the lamp via a Mobile App. Morse alphabet:

Tell the participants about the history of telecommunication. You can use the infographic attached (annex 3).

We will use a free on-line tool from the *https://appinventor.mit.edu/* website. A Google account is needed to log in. You can prepare a few accounts before. Click **Create Apps!**, log in and create a new project (**Start new project**). Ask the participants to install the MIT App Inventor 2 application on their smartphones (with Android).

The view is divided into a few sections (Fig. 17): **User Interface**, **Viewer**, **Components** and **Properties**. All above sections can be found in the Designer tab. From the **User Interface** you can choose elements of the application and move them to the **Viewer** section. In this section you can see how the application looks. Some components are visible (e.g. buttons), some are hidden (e.g. a voice message after clicking on a button). In the **Properties** section you can change options of the components.

In the **Blocks** tab you can create a block program in a similar way to the TUNIOT.

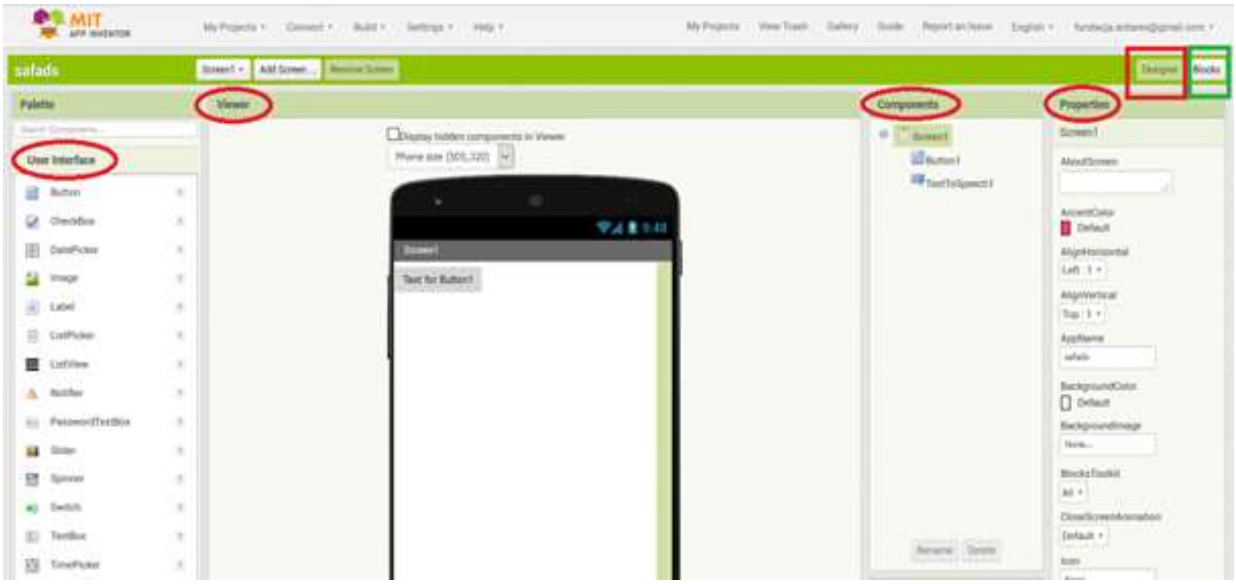


Fig. 17. Interface of the MITapp program.

Exercise 10 (optional): Our first application consists of a button which plays a voice message. From the User Interface choose the Button component and drop it in the **Viewer**. The next component **TextToSpeech** can be found in the **Media**. It is a hidden component so it is not displayed in the **Viewer**, but under the picture of a smartphone. In the **Blocks** tab choose **when Button1.Click do** from the **Button1** menu. Stick **call TextToSpeech1.Speak message** block from the **TextToSpeech1** menu to it. The message should be written in a text block (first block in the **Text menu**) and stuck to the rest of the code (Fig. 18).

Fig. 18. A block code to the exercise 10.

From the very upper menu select **Connect -> AI Companion**. An QR code should show on the screen. You can scan it with the MIT App Inventor 2 application or type a 6-digit code in the mobile application. Test your application. Remember that the mobile phone and the computer should be in the same local network.

Exercise 11: In this exercise we will control our diodes via a mobile app. Come back to the **Designer** view. Add the second button and change the names of the buttons for 'ON' and 'OFF'. You can do it by selecting the **Button1** in the **Components** menu and changing the **Text** field in the **Properties** menu (marked with a blue circle at Fig. 19). Add one more hidden component: **Web** from the **Connectivity** menu. Your **Designer** view should look like at Fig. 19.

In the **Blocks** view delete the purple and red block if you completed the Exercise 10. If you skipped Exercise 10, add the **when Button1.Click do** block from the **Button1** menu. Stick to it the **set Web1 Url to** block (from **Web1** menu) and an empty text block from the **Text** menu (it is the first block in this menu). Copy your IP address to the empty text block. After the adres write '/ON' text. So the text in the text block should be e.g.: http://192.169.136.80/ON. Choose the **call Web1.Get** block from the **Web1** menu and stick it below. This part of the code will change the text at the IP address for 'ON' after the first button is clicked.

Copy-paste the whole code and change the 'Button1' for the 'Button2' in the most external block. Change also the 'ON' text after your IP address for 'OFF'. Ask your participants if they can guess how the second part of the code works. Upload your code in the MIT App Inventor 2 application as in the exercise 10 (dark red text) and test it.

Fig. 19. MITapp interface during the preparation of exercise 11.

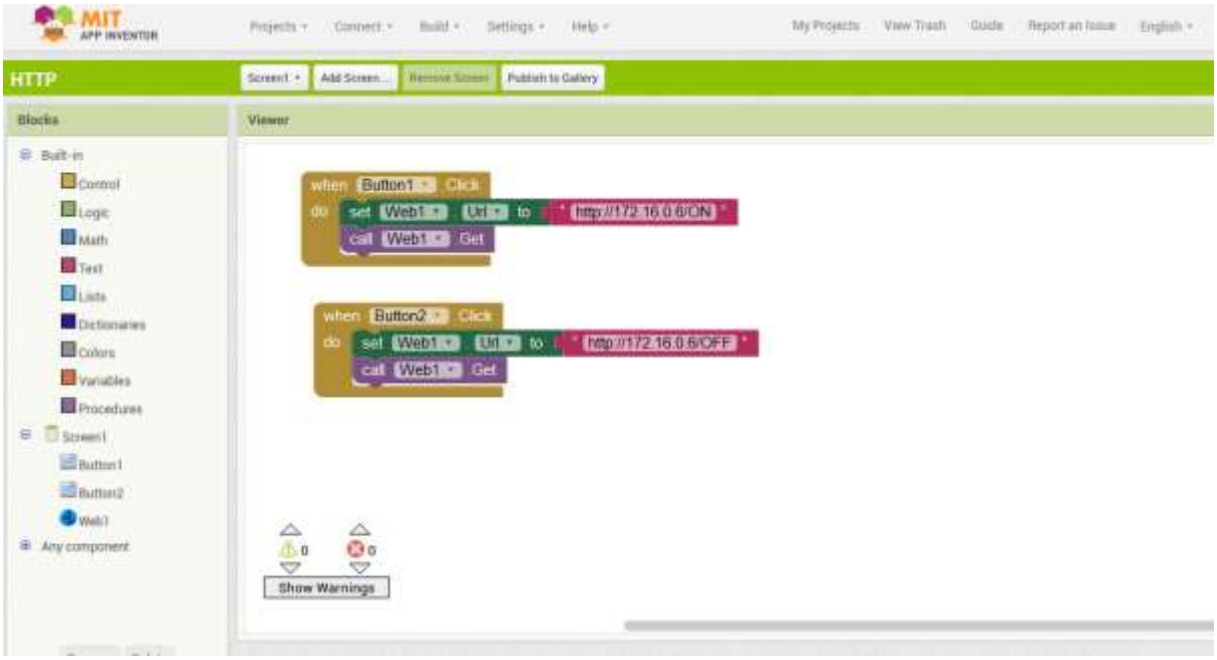


Fig. 20. A block code to the exercise 11.

4. Closing:

Using sensors. Work in groups.
Each group gets pictures with a few sensors (cut out from the Annex 1). Ask the participants to discuss in groups possible usage of those sensors in daily life/industry/communication. At the end of exercise each group presents its ideas.

Workshop evaluation. Discussion.

Ask the participants what their impressions after the first class is. What was the greatest difficulty/challenge? What do they remember about the Internet of Things? Can they explain the idea?

Troubleshooting:

- Compilation problem: path to the Arduino folder is too long (move it to the C:)
- Wrong port: in arduino-1.8.1 select Tools -> Port -> COMx (usually the last one)
- Do not plug something else to the other USB port (e.g. a telephone, a mouse)
- Computer should be plugged to the power
- Check components if they are working
- Check USB cable – cheap ones can break quickly
- Check wiring: if the ground PIN is connected
- Check code: did you choose the correct PIN; common mistake – forgetting about delays
- Do not forget to delete all the old blocks in the code
- Serial Monitor should be closed before executing the next program

Annex 1:

barometric pressure sensor

motion sensor

slope sensor

infrared sensor

pressure sensor

humidity sensor

PH sensor

fingerprints reader

fluid flow sensor

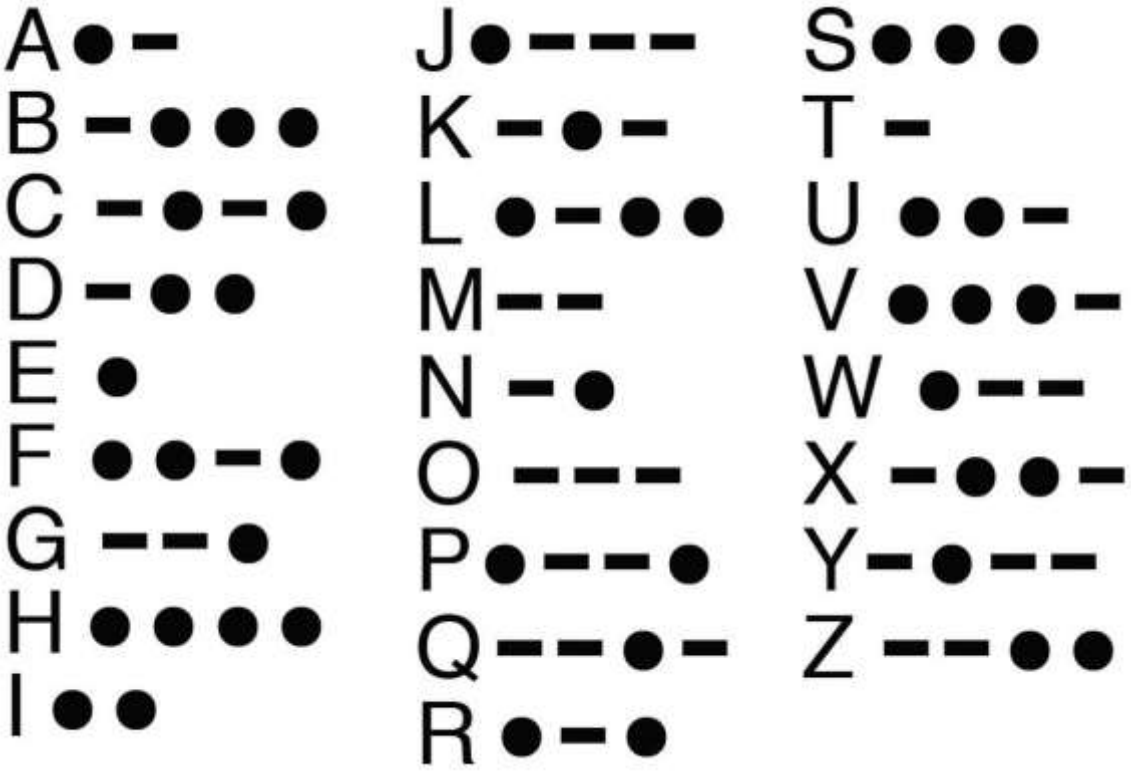
sound detector

light and colour detector

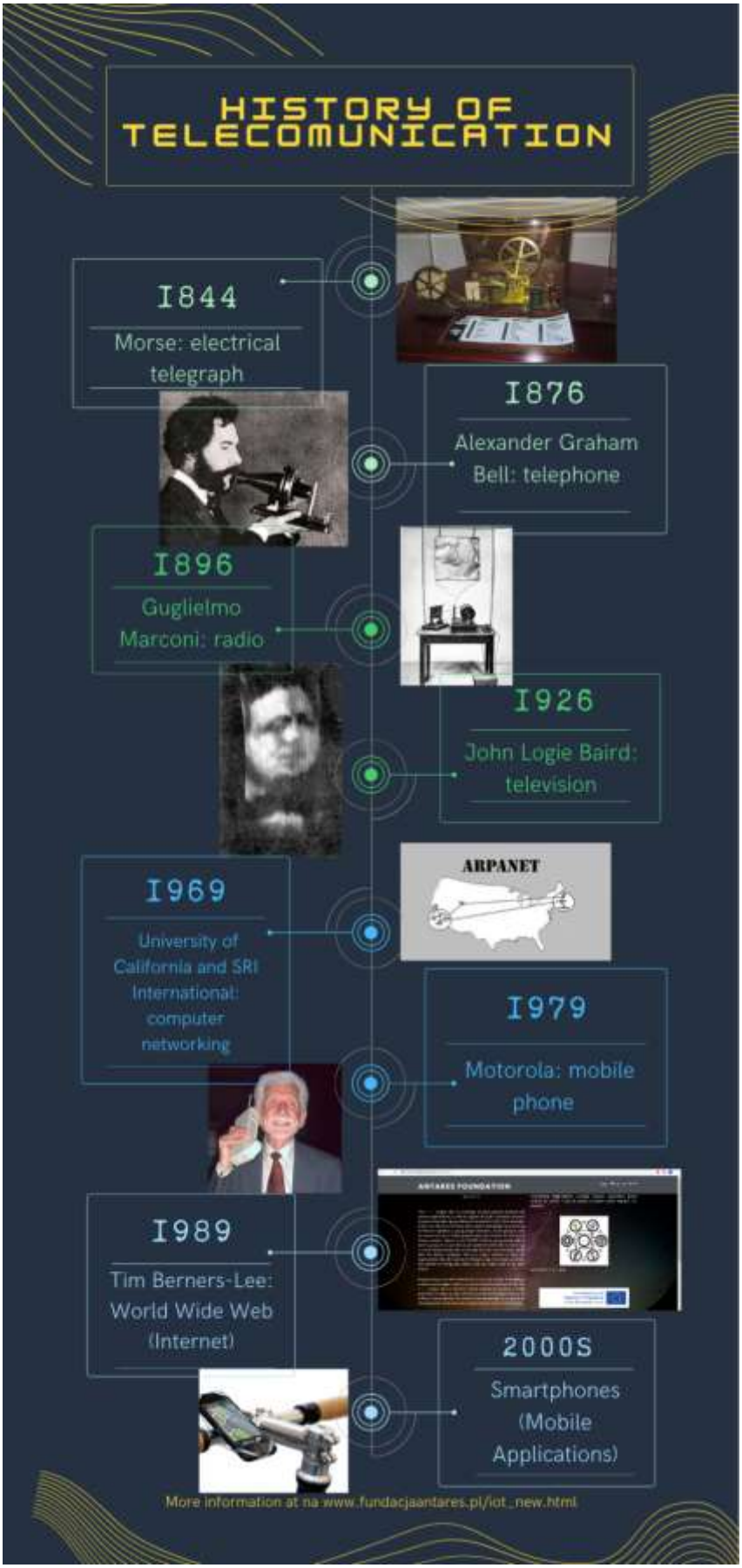
temperature sensor

accelerometer

Annex 2:



Annex 3:


HISTORY OF TELECOMUNICATION
1844
Morse: electrical telegraph
1876
Alexander Graham Bell: telephone
1896
Guglielmo Marconi: radio
1926
John Logie Baird: television
ARPANET
1969
University of California and SRI International: computer networking
1979
Motorola: mobile phone
1989
Tim Berners-Lee: World Wide Web (Internet)
2000S
Smartphones (Mobile Applications)
More information at na www.fundacjaantares.pl/iot_new.html