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

Task sheets from Module A: Microcontrollers

#	Handout	#	Task sheet
A1	Basic introduction to Microcontrollers	**-**	N/a
A2	Logic Gates	**A2.1**	Logic Gates (AND, OR)
		A2.2	Logic Gates (AND, NOT)
		A2.3	7-segment Display Counter
A3	Introduction to (Block) coding	**A3.1**	Programming a click counter
		A3.2	Programming a mini piano
		A3.3	Tilt the microcontroller to light the LED up
		A3.4	Programming the game “Rock, paper, scissors”
		A3.5	Mental arithmetic vs. a microcontroller
		A3.6	Programming the game “Hot Potato”
		A3.7	Programming the game “Avoid the obstacle”
		A3.8	Programming a game “Catch the LED”
		A3.9	Block coding practice with a pre-made circuit
A4	Arduino Basics	**A4.0**	Learning if/else-query and while-loop
		A4.1	Controlling a LED using a push button
		A4.2	Connecting a PIR sensor to an Arduino
		A4.3	Connecting an ultrasonic sensor to Arduino
		A4.4	Creating an air conditioning system from scratch
A5	Introduction to IoT	**A5.1**	Arduino weather station
		A5.2	Create your local server
		A5.3	Sensing the Environment & Notifying

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

Module: Microcontrollers

Topic: Logic Gates

Task sheet A2.1: Logic Gates (AND, OR)

Time: 2:00 hours

General description:

The input/output information of any logic gate or circuit can be represented through a standard truth table. In this activity, participants will be asked to discover the output signal (OUT), given Signal 1 (S1), Signal 2 (S2) and Signal 3 (S3) and write it on the truth table given below according to the logic gate diagram. Afterwards, participants will test their design using a smartphone app.

Learning objective(s):

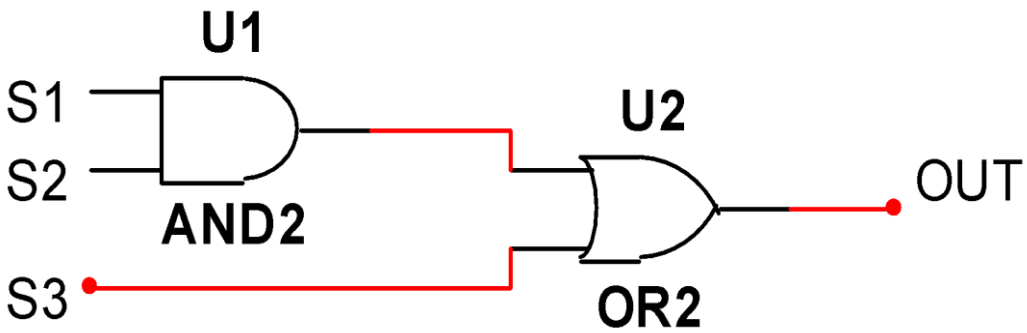
- Identify and test logic gates operations AND and OR.
- Test a logic design and debug it.

Material required:

- Smartphone with Logic Gate Circuit game;
- Pen and paper to organise the truth table.

Description of the activity:

1. Analyse the logic gate diagram given below:



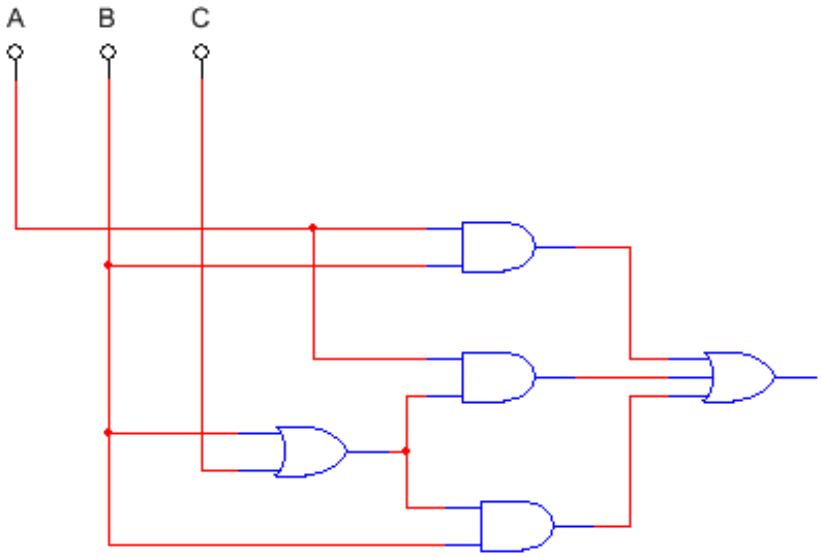
2. Afterwards, fill in the truth table with the Output information:

S1	S2	S3	OUT
0	0	0	
0	1	1	
1	0	0	
1	1	1	

3. By using a Logic Gate app, test the design plotted.

How to adapt to different learners:

- Participants with more knowledge can create diagrams that are more complex and plot the correspondent truth tables. Other logic gates (such as XOR, NOT, NAND, NOR, XNOR) may also be used. Example of a diagram that they can plot:



- Participants should be allowed to play the Logic Gate game (e.g. Circuit Scramble APK; Logic Gate Simulator; Smart Logic Simulator; Logic Circuit Simulator Pro) for at least 20 to 30 minutes, so that they can test and debug other logic gate diagrams.

Additional information:

- More examples of truth tables using different logic gate combinations:

AND

AND

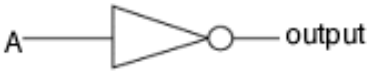
A	B	Output
0	0	0
1	0	0
0	1	0
1	1	1

OR

OR

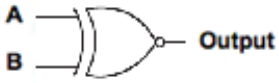
A	B	Output
0	0	0
1	0	1
0	1	1
1	1	1

NOT



A	output
0	1
1	0

XNOR

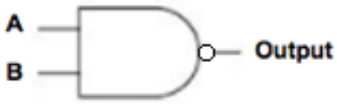


A	B	Output
0	0	1
1	0	0
0	1	0
1	1	1

XOR

A	B	Output
0	0	0
1	0	1
0	1	1
1	1	0

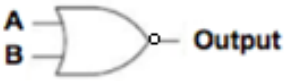
NAND



NAND

A	B	Output
0	0	1
1	0	1
0	1	1
1	1	0

NOR



NOR

A	B	Output
0	0	1
1	0	0
0	1	0
1	1	0

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

Module: Microcontrollers

Topic: Logic Gates

Task sheet A2.2: Logic Gates (AND, NOT)

Time: 2:00 hours

General description:

The input/output information of any logic gate or circuit can be represented through a standard truth table. In this activity, participants will be asked to discover the output signal (OUT), given Signal 1 (S1), Signal 2 (S2) and Signal 3 (S3) and write it on the truth table given below according to the logic gate diagram. Afterwards, participants will test their design using a smartphone app.

Learning objective(s):

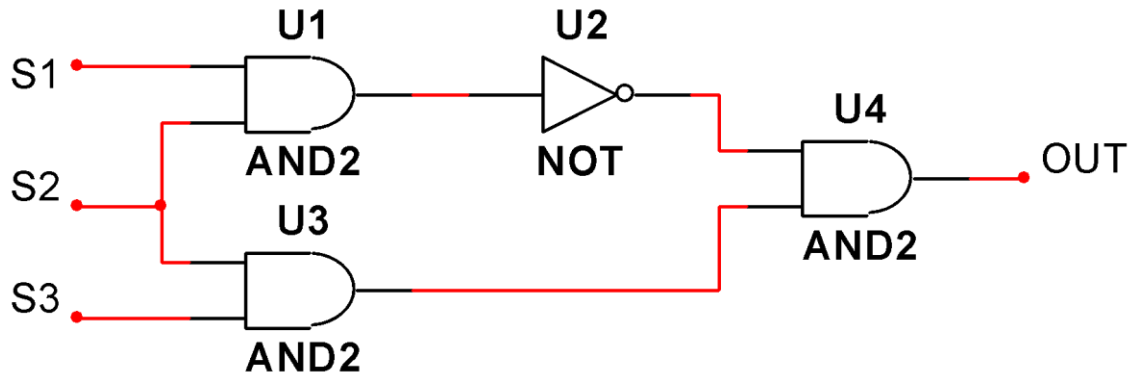
- Identify and test logic gates operations AND and NOT.
- Test a logic design and debug it.

Material required:

- Smartphone with Logic Gate Circuit game;
- Pen and paper to organise the truth table.

Description of the activity:

1. Analyse the logic gate diagram given below:



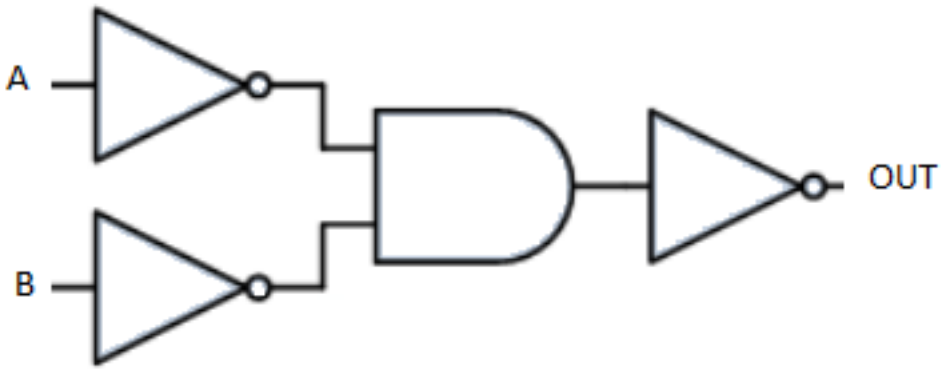
2. Afterwards, fill in the truth table with the Output information:

S1	S2	S3	OUT
0	0	0	
0	1	1	
1	0	0	
1	1	1	

3. By using a Logic Gate app, test the design plotted.

How to adapt to different learners:

- Participants with more knowledge can create diagrams that are more complex and plot the correspondent truth tables. Other logic gates (such as XOR, NAND, NOR, XNOR) may also be used. Example of a diagram that they can plot:

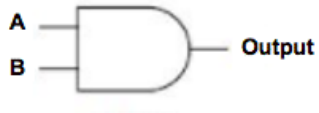


- Participants should be allowed to play the Logic Gate game (e.g. Circuit Scramble APK; Logic Gate Simulator; Smart Logic Simulator; Logic Circuit Simulator Pro) for at least 20 to 30 minutes, so that they can test and debug other logic gate diagrams.

Additional information:

- More examples of truth tables using different logic gate combinations:

AND



AND

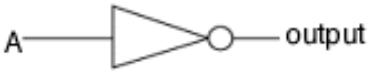
A	B	Output
0	0	0
1	0	0
0	1	0
1	1	1

OR

OR

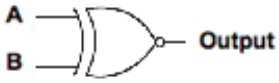
A	B	Output
0	0	0
1	0	1
0	1	1
1	1	1

NOT



A	output
0	1
1	0

XNOR

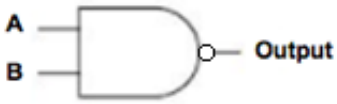


A	B	Output
0	0	1
1	0	0
0	1	0
1	1	1

XOR

A	B	Output
0	0	0
1	0	1
0	1	1
1	1	0

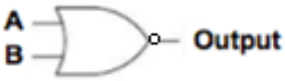
NAND



NAND

A	B	Output
0	0	1
1	0	1
0	1	1
1	1	0

NOR



NOR

A	B	Output
0	0	1
1	0	0
0	1	0
1	1	0

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

Module: Microcontrollers

Topic: Logic Gates

Task sheet A2.3: 7-segment Display Counter

Time: 2:00 hours

General description:

A 7-segment display is a device that consists of 7 LEDs that allows to display numbers based on the different LED configurations. They are responsible for showing the numbers on many electronic devices, such as digital clocks and basic calculators. In this task, participants will use a 7-segment display with a 1-digit counter that will be able to display numbers from 0 to 9. The task can be done physically or using a software simulator.

Learning objective(s):

- Be able to identify different chips and components
- Be able to build a seven-segment display counter.

Material required:

Physical Assembly:

- Breadboard
- Jumper cables
- Seven segment decoder (7448 chip)
- Synchronous 4-bit up/down counter (74192 chip)
- Precision timer (Ne555 chip)
- Power supply (5v rated)
- 330Ω resistor
- 1.8kΩ resistor
- 10kΩ resistor
- 10uF capacitor
- Seven-segment displays (common cathode)

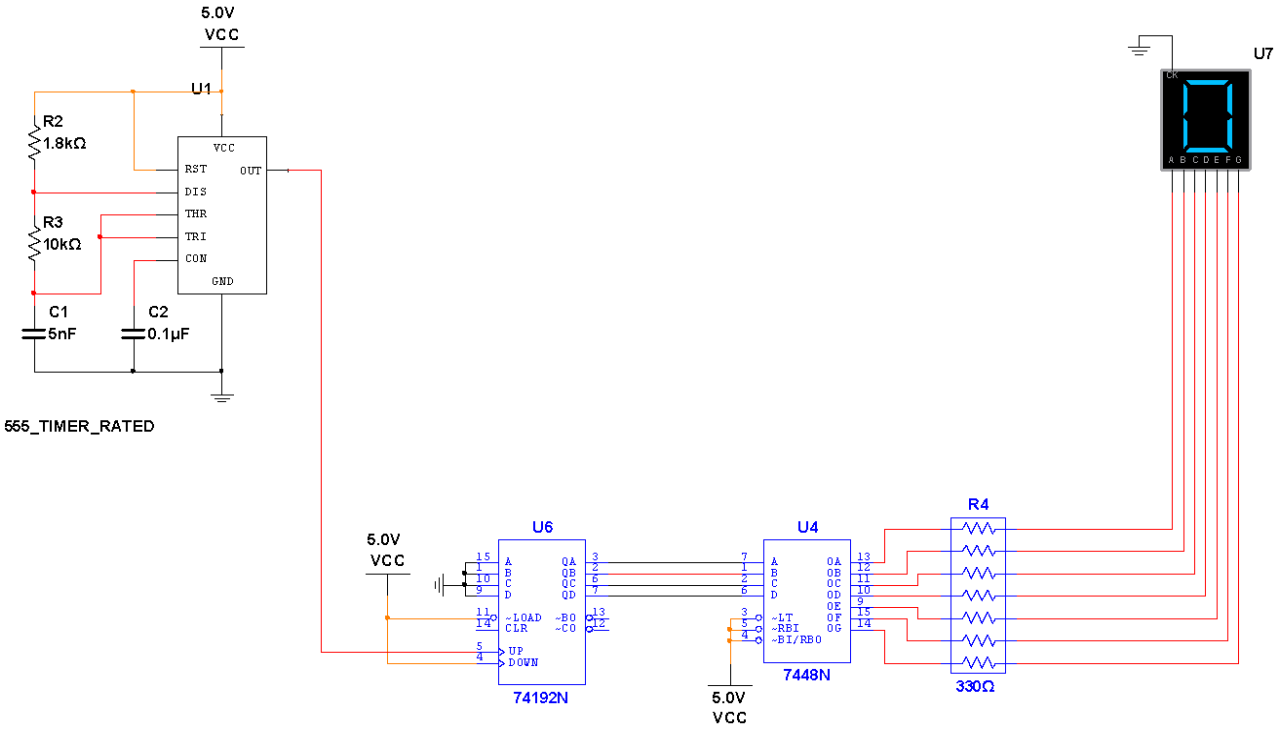
Or

Software Assembly:

- Multisim (student version)

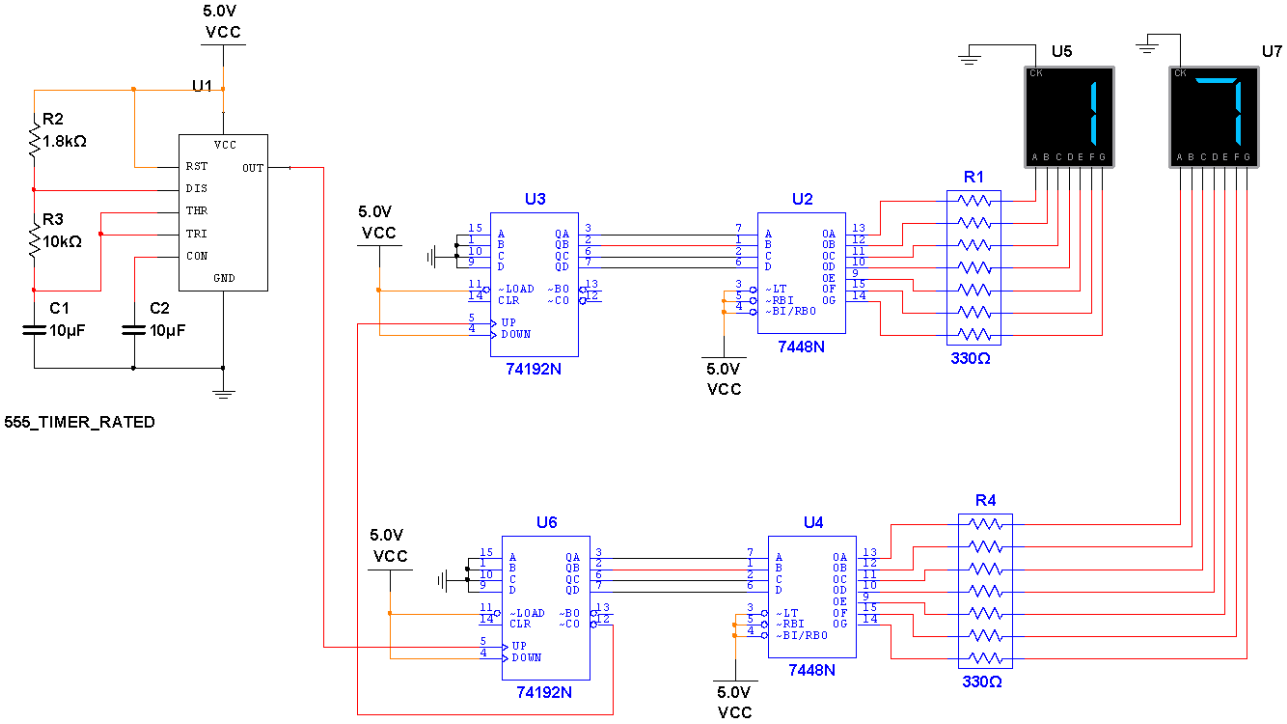
Description of the activity:

1. Connect the materials required following the diagram below:



How to adapt to different learners:

- It is important that participants understand the different chips and components that are going to be used. Therefore, have them do research on these topics and then encourage an open discussion to know their findings.
- Other assembly configurations can be performed by more advanced participants:



Additional information:

Tutorials on creating a 7- segment display counter:

- https://www.youtube.com/watch?v=WEQeqLyWN3s
- https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html

To learn more about the used chips:

- NE555: xx555 Precision Timers datasheet (Rev. I)
- 74192:https://datasheetspdf.com/pdf-file/248168/STMicroelectronics/74192/1
- 7448: https://datasheetspdf.com/pdf-file/1407971/etcTI/7448/1

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

Module: Microcontroller

Topic: Introduction into (Block)Coding

Task sheet A3.1: Programming a click counter

Time: 00:15 hour(s)

General description:

In this task sheet we want to programme a simple click counter which counts how many times a button was pressed and show it on the LED matrix.

Learning objective(s):

The participants learn how to:

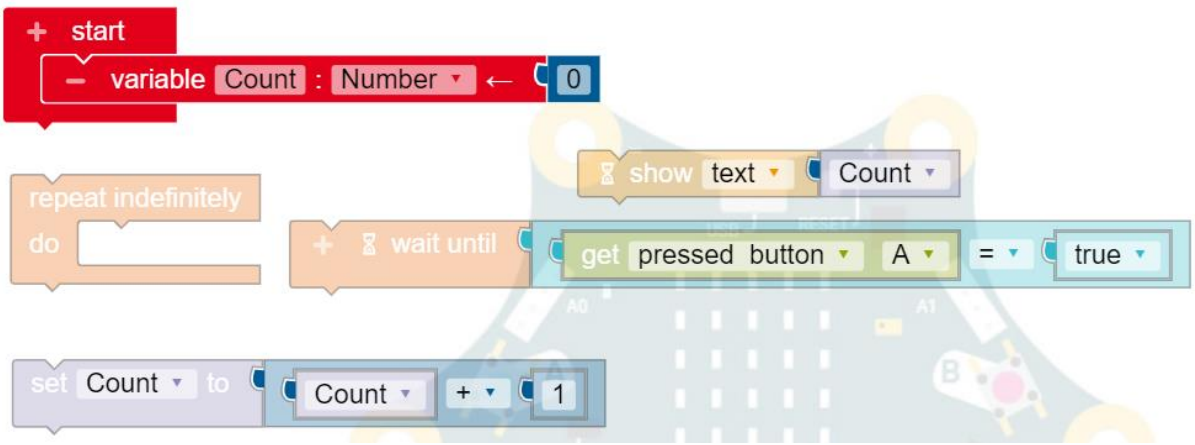
- use a loop:
- implement a button and let the microcontroller wait until the button is pressed:
- create and manipulate a variable:
- use the LEDs.

Material required:

- Microcontroller of your choice (Calliope mini was used here)
- Laptop

Description of the activity:

You need a variable first. Click on the plus in the start block. Change the name of the variable to "Count". Then use these commands:



How to adapt to different learners:

The task is very basic and is only meant to give the participants a general knowledge of programming but it can be expanded easily by the participants. For example, you can now change the programme so that it only adds 1 to “Count” when both buttons are pressed (if there are two buttons) or you can give out a sound every time the button is pressed.

Additional information:

- You can find additional tasks in this handout: Module: Microcontroller - Topic: (Block)Coding

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

Module: Microcontroller

Topic: Introduction into (Block)Coding

Task sheet A3.2: Programming a mini piano

Time: 00:15 hour(s)

General description:

In this task sheet the participants will develop a programme using any microcontroller with a buzzer and a button. The programme is going to be a “mini piano”. By pressing a button, a sound will be produced.

Learning objective(s):

The participants learn how to:

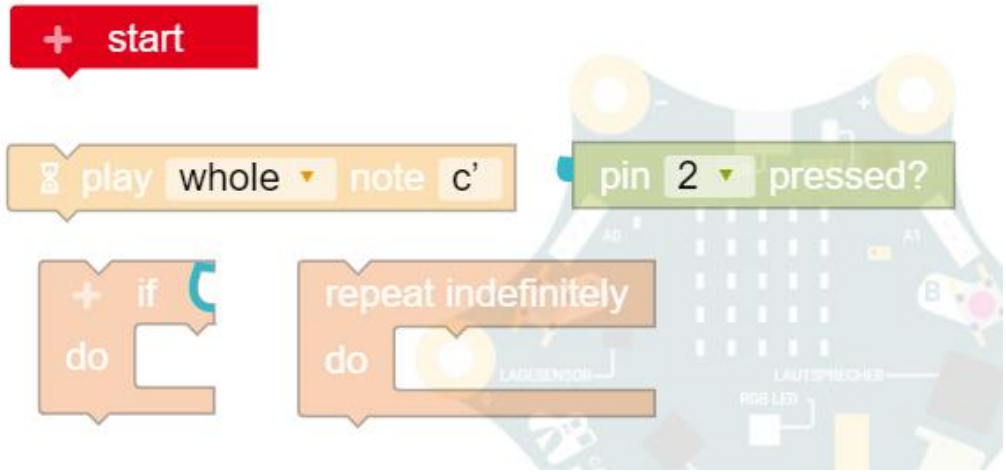
- use a loop
- implement a button/pin usage
- use a if query
- use sound

Material required:

- Microcontroller of your choice (Calliope mini was used here)
- Laptop

Description of the activity:

Use these blocks to create the programme:



How to adapt to different learners:

The task is very basic and is only meant to give the participants a general knowledge of programming but it can be expanded easily by the participants. For example, you can now change the programme so that it plays another sound or you can add more buttons/pins and let the microcontroller play another sound by pressing them. This way you get a “mini piano”.

Additional information:

- You can find additional tasks in this handout: Module: Microcontroller - Topic: (Block)Coding

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

Module: Microcontroller

Topic: Introduction into (Block)Coding

Task sheet A3.3: Tilt the microcontroller to light the LED up

Time: 0:15 hour

General description:

In this task sheet we want to programme a simple programme that lights up the LED as soon as the microcontroller is upside down. We use a calliope mini and the open-roberta lab to develop the programme.

Learning objective(s):

The participants learn how to

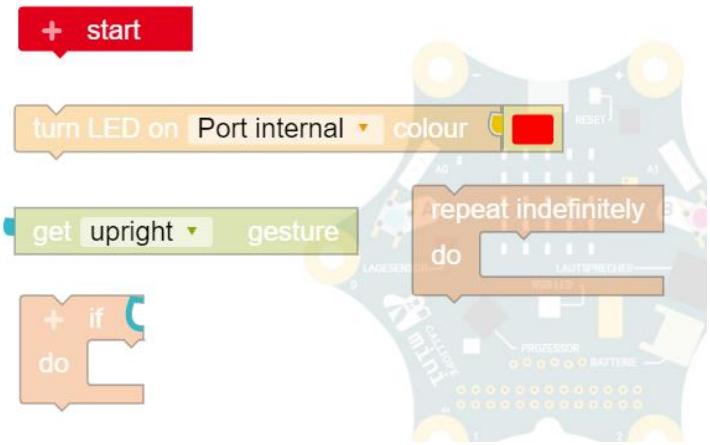
- use a loop
- use the gyroscope
- use a if-query
- turn a LED on and assign a colour to it

Material required:

- Microcontroller of your choice (Calliope mini was used here)
- Laptop

Description of the activity:

Let the LED light up in green when the Calliope mini is upright. When the Calliope mini is upside down, let the LED glow red. Use these commands for this:



How to adapt to different learners:

The task is very basic and is only meant to give the participants a general knowledge of programming but it can be expanded easily by the participants. For example, you can now change the programme so that the LED gets a different colour or so that several LEDs light up when the microcontroller is upside down. Furthermore, you can let a second LED of any colour light up when the microcontroller is up right.

Additional information:

- You can find additional tasks in this handout: Module: Microcontroller - Topic: (Block)Coding

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

Module: Microcontroller

Topic: Introduction into (Block)Coding

Task sheet A3.4: Programming the game “Rock, paper, scissors”

Time: 0:10 hour

General description:

In this task sheet we want to programme a little game called “rock, paper, scissors” with a micro:bit. This task sheet is a text only task sheet and has no visual help.

Learning objective(s):

The participants learn how to

- use a loop
- use a if-query
- create and manipulate a variable
- use the LEDs

Material required:

- Microcontroller of your choice (micro:bit was used here)
- Laptop

Description of the activity:

Task 1 - The basics

In order to have the option to choose, the symbols must first be taught.

a) Use LEDs to paint the stone, the scissors and the paper.

b) By shaking the micro:bit the selection should take place.

c) After the shaking is detected a variable should be set to a random number corresponding to the number of symbols.

Task 2 - The query

At the moment we only have the symbols and a variable with a random number. Now the symbols have to be assigned to a number.

How to adapt to different learners:

The task is very basic and is only meant to get the participants away from the given structures in the previous task sheets. This task sheet is only in text and not with any blocks.

Additional information:

- You can find additional tasks in this handout: Module: Microcontroller - Topic: (Block)Coding

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

Module: Microcontroller

Topic: Introduction into (Block)Coding

Task sheet A3.5: Mental arithmetic vs. a microcontroller

Time: 00:15 hour(s)

General description:

In this task sheet we want to programme a microcontroller in a way that it does arithmetic. It shows the first number, then the operand and then the second number on its LED matrix. After a short pause the solution is shown on the LED matrix.

Learning objective(s):

The participants learn how to;

- use a loop
- implement a button and let the microcontroller wait until the button is pressed
- create and manipulate a variable by assigning randomized numbers to it
- use the LEDs
- use the wait function
- generate a variable that is calculated by using two other variables and an arithmetic operand

Material required:

- Microcontroller of your choice (Calliope mini was used here)
- Laptop

Description of the activity:

Task 1 - Create and show the numbers

First of all, we need two numbers to calculate.

a) Create two variables and name the numbers as you like. The two variables get the value 0.

b) Let the Calliope mini display the two numbers in an endless loop.

Task 2 - Bringing the keys into the game

a) Only display the numbers when the A button is pressed.

b) If button A is pressed, the Calliope should assign a random value between 1 and 10 to the created numbers. The change takes place before the numbers are displayed.

c) Display the first number after the change. Then display your desired arithmetic operation (+ , - , : , *). Now display the second number.

Task 3 - Do you calculate correctly?

At the moment the arithmetic problem is only being displayed. Now the Calliope should calculate itself and show you the result!

a) So that the calculation duel remains fair you need some time. Build in a waiting time (1000ms = 1s) after the display of the arithmetic task.

b) After the waiting time has elapsed, let the result of the calculation be displayed as text.

For this the Calliope must calculate the task first! Implement the calculation.

How fast can you calculate?

By shortening the waiting time, you can make the calculation duel more exciting and put your head calculation skills to the test!

How to adapt to different learners:

This task is an intermediate task. The participants do not work with given blocks anymore but have to create their own logic and have to implement a little structure by themselves (though a structure is given by the tasks).

Most of the participants will have problems with that. However, some participants may finish early. You can keep these participants busy by letting them programme a way where the operand is randomised by creating a list of operands.

If this is too hard for the participants you can let them expand the programme by adding another number and a second operand.

If someone really is an expert you can let him/her programme a randomised mental arithmetic task that contains brackets and several numbers and operands.

Additional information:

- You can find additional tasks in this handout: Module: Microcontroller - Topic: (Block)Coding

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

Module: Microcontroller

Topic: Introduction into (Block)Coding

Task sheet A3.6: Programming the game “Hot Potato”

Time: 1:00 hour

General description:

We want to programme a game called “Hot Potato”. In this game the microcontroller is passed from player to player after a button is pressed to start the game. Each player can decide how long he/she wants to hold the “Hot Potato”. The game ends when the time is up and the player with the “Hot Potato” in his/her hands loses the game. A Calliope Mini was used but it works with every microcontroller that has buttons, LEDs (and sound). For this we use the "expert blocks" of the open-roberta editor. These are activated by clicking on ☆2.

Learning objective(s):

The participants learn how to:

- use a loop
- implement a button and let the microcontroller wait until the button is pressed
- create and manipulate a variable
- use the LEDs
- use sound
- use a wait function
- use a if-query

Material required:

- Microcontroller of your choice (Calliope mini was used here)
- Laptop

Description of the activity:

Task 1 - Creating and changing a variable

The A key on the Calliope Mini is used to determine the start signal.

a) Create a variable and assign an integer value to it.

b) If key A is pressed, this variable should be reduced by a random, small value.

c) Let the LED of the Calliope Minis light up as soon as the number has been changed.

Task 2 - First Wait and First Tone

A variable has now been created that has been changed. Now a waiting time shall be added to make the game more random.

a) Install a block which makes the Calliope Mini wait. First you shall wait randomly between 300 and 500ms.

b) Change the programme so that the variable from task 1 is changed when you press the A key. Then wait randomly between 300 and 500ms and finally light up the LED.

c) In addition to the LED lighting, make a sound. Frequency and length are up to you.

Task 3 - The loop

The game was made more random by task 2. But so far it is only one step sequence long. The game shall also work in a loop.

a) Select a loop block that runs through a loop until a certain condition is reached.

b) How do I change the programme to run it in a loop after pressing the A key? Make these changes to your programme.

c) Which condition would be useful as an abort condition? Programme this abort condition for the loop block.

Task 4 - The end of the game

The end of the game should now be signaled by the previously programmed lighting up of the LED and the sound output.

a) Change your programme so that only the variable is changed and maintained in the loop block.

b) Let the tone sound and the LED light up as soon as the loop block abort condition is reached.

How to adapt to different learners:

This task is more complex and most participants will need the time given. However, if some participants finish the task faster, you can give the following tasks:
At the moment, the game is only played once. It would be better if the game can be restarted each time by pressing the B key. Furthermore, optical signals (pictures, LED) during the game would certainly be a good extension. If you already finished your tasks you can try out the following tasks.

a) How would it be possible to restart the game again and again with button B?

b) How would it be possible to output optical signals during the game? An example would be smileys (smiling during the game / sad as soon as the game is over).

Additional information:

- You can find additional tasks in this handout: Module: Microcontroller - Topic: (Block)Coding

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

Module: Microcontroller

Topic: Introduction into (Block)Coding

Task sheet A3.7: Programming the game “Avoid the obstacle”

Time: 1:00 hour

General description:

In this task sheet we programme a game where the player has to avoid obstacles with his player LED. The obstacles are LEDs that move downwards while the player LED is on the bottom and can move right and left to avoid these obstacles.

Learning objective(s):

The participants learn how to:

- use a loop
- implement a button and let the microcontroller wait until the button is pressed
- use a sprite
- create and manipulate a variable
- use the LEDs
- use a if-query
- use the wait function
- use a while loop

Material required:

- Microcontroller of your choice (micro:bit was used here)
- Laptop

Description of the activity:

A hint in advance:

It does not always have to be the case that there is only one "permanent" block. A programme can also have several!

Task 1 - Initialise the game

First of all, the basics are to be set up. This time, however, this is done in a "permanent" block.

a) Create three variables named as follows: "score", "playerLED", "alive".

The "score" is set to 0. Your "playerLED" will be a sprite at position x=2 and y=4. The variable "alive" will be set to true.

b) Build a loop, which takes a break, if the player is alive.

c) After the loop the score is to be determined by the variable "score".

d) After the score has been set, the game is finished.

Task 2 - The keys to move

The player LED moves to the left or right using the A and B buttons. For this the following tasks have to be done. Please note that the player LED should not leave the edge of the game.

a) If the key A is pressed, it should be checked whether the player LED is at the edge of the game. When is this the case?

b) If the player LED is not at the edge of the game, the x-coordinate of the LED should be reduced by one.

c) The same must be done for the B-button. The LED will move in the other direction.

Task 3 - The first obstacle

Your LED can now move and the basics of the game are created. Now the game still needs obstacles! The obstacle (an LED) will appear at the top of the screen and will wander down. At the end it will be checked if the obstacle collided with your player LED. If this is not the case, the obstacle is simply brought back to the upper position with a random delay.

a) In another "permanent" block, a pause is taken first to make the game playable. Furthermore, a variable is needed to represent the obstacle.

b) Check if you are alive and if so, the first obstacle should be a sprite at x=0 and y=0. After that set up a randomly large pause.

c) Now a loop comes into the query. The loop should query the following while you are alive:

- If the obstacle has the y-coordinate = 4 you have to check if the player LED touches the obstacle or not. (Nested query)
- If the player touches the obstacle with his LED, the variable "alive" must be set to wrong. Otherwise your score should be increased by one and the obstacle should be moved back to its original position with a randomly large pause (which coordinate has to be changed for this?).
- If the obstacle is not at the position y = 4, the y-coordinate of the obstacle is increased by one. After this, a small pause is taken.

d) Only one obstacle exists so far. However, it is relatively simple to build further obstacles out of this obstacle. What must be done for this?

How to adapt to different learners:

This task is more complex and most participants will need the time given. Weaker participants may need help. The biggest problems occurred in the part where the LEDs have to get moving. Explain this with a grid (coordinate system) since the LED matrix works exactly like this. (Example: LED has to move to the right → x-Axis variable has to be increased by 1. Or LED has to move down → y-Axis variable has to be reduced by 1)

For participants who finished the task you can give the following tasks:

- Change the code so that the obstacles get faster the higher your score is.
- Add sound effects when the player gets points/ loses the game.

Additional information:

- You can find additional tasks in this handout: Module: Microcontroller - Topic: (Block)Coding

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

Module: Microcontroller

Topic: Introduction into (Block)Coding

Task sheet A3.8: Programming a game “Catch the LED”

Time: 1:00 hour

General description:

In this task sheet we want to programme a little game where the player has to catch a sideways moving LED when it’s in the middle of the LED matrix. The LED is caught by pressing a button.

Learning objective(s):

The participants learn how to:

- use a loop
- implement a button and let the microcontroller wait until the button is pressed
- use a sprite
- create and manipulate a variable
- use the LEDs
- use a if-query
- use the wait function
- use a while-loop

Material required:

- Microcontroller of your choice (micro:bit was used here)
- Laptop

Description of the activity:

Task 1 - A sprite please!

To create a moving LED you need a so-called "sprite".

A sprite is nothing more than a graphic object. An example of a Sprite is the game character Super Mario.

a) Create a variable called "sprite".

b) Change the variable and create a sprite in the middle of the LED field.

Task 2 - Bringing Movement into the Game

The sprite is now displayed in the middle of the field. Now you have to bring some movement in!

a) Now let the sprite move by changing your variable permanently by one.

b) The sprite must now be able to bounce off the edge. Build such a block after the movement.

c) The sprite should be able to walk back again. For this it must be able to bounce off the second edge.

Task 3 - Catch the LED

Last but not least, there must be a way to catch the sprite.

a) Build in a way to catch the sprite with the A button. Use an input block for this!

b) Now you have to ask if the sprite is really in the middle. If so, your score should be increased by 1. If not, the game should be ended.

How to adapt to different learners:

This task is more complex and most participants will need the time given. Weaker participants may need help. The biggest problems occurred in the part, in which the LEDs are supposed to move. Explain this with a coordinate system since the LED matrix works exactly like this. (Example: LED has to move to the right → x-Axis variable has to be increased by 1.)

- Let the LED move faster every time you catch it (or just faster in general)
- If you lose, the microcontroller should play a sound and display a sad smiley
- Every time you catch the LED a sound should be played
- Display your score as soon as you lost

Additional information:

- You can find additional tasks in this handout: Module: Microcontroller - Topic: (Block)Coding

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

Module: Microcontrollers

Topic: Introduction into (Block)Coding
Task sheet A3.9: Block coding practice with a pre-made circuit

Time: 3:00 hours

General description:

One of the essential things when starting using Arduino is to get used to the coding language. This activity will allow the participants to create diverse codes using block coding in order to get introduced to the programmatic way of thinking. This activity is thought to be made using the developing environment mBlock in order to help the participants with low coding skills get into the coding language. The trainer can print each exercise separately, so to give the next "challenge" only when the previous one is done correctly.

Learning objective(s):

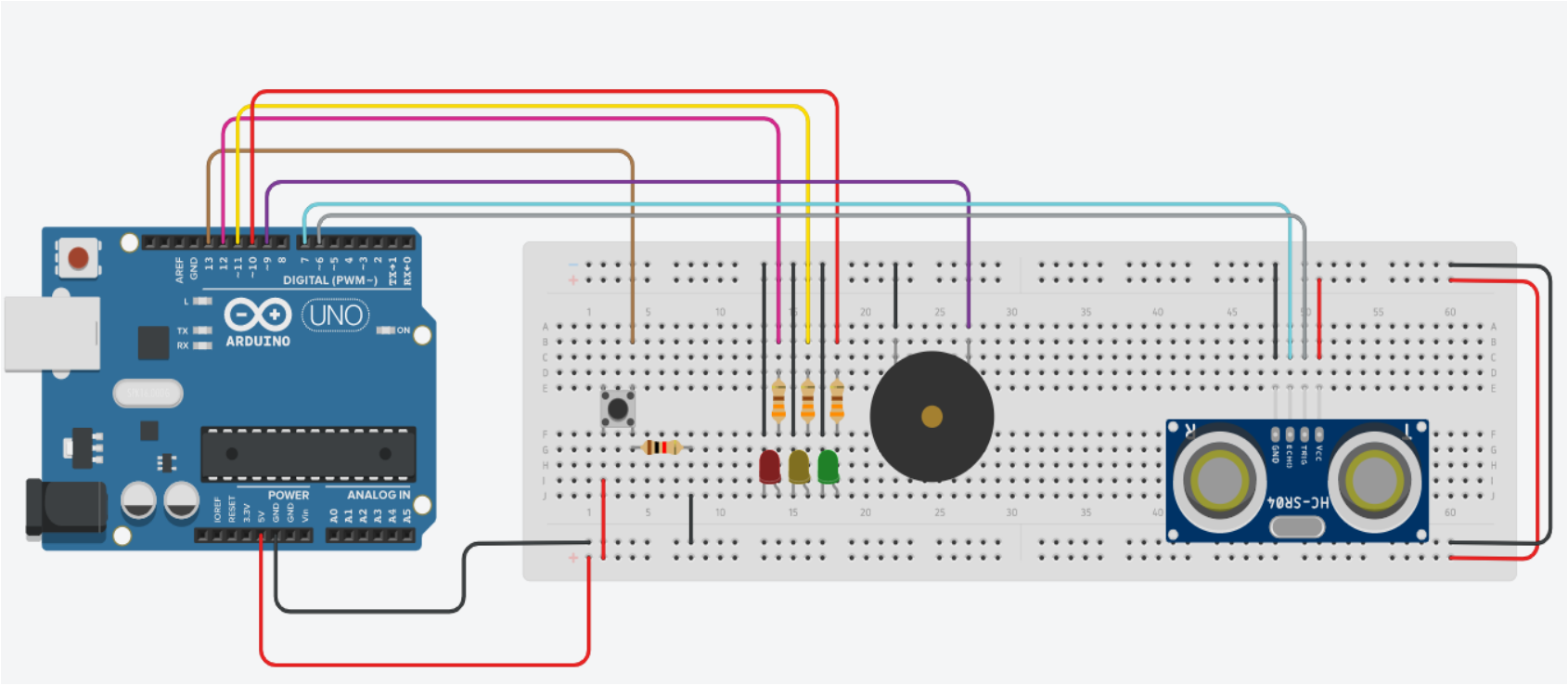
The participants learn how to:

- Implement the code to turn ON and OFF a LED light;
- programme a traffic light;
- implement the code to use a buzzer;
- implement a button and let the microcontroller wait until the button is pressed;
- implement the code to control an ultrasound sensor and read the input value

Material required:

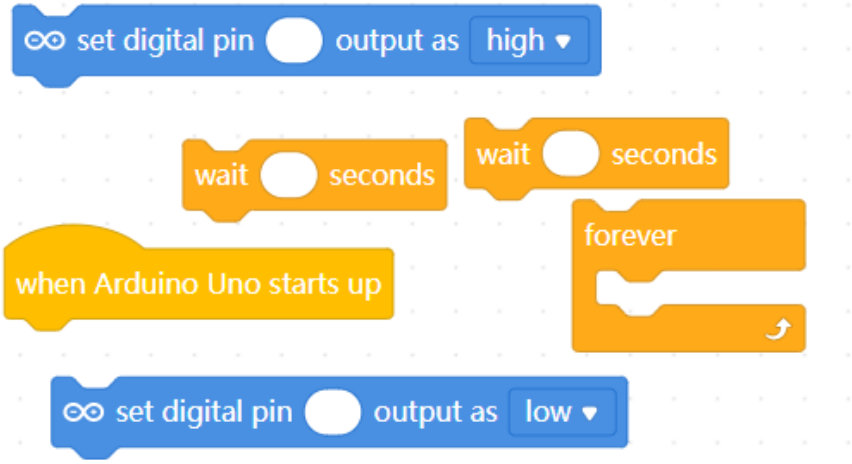
- Laptop with access to Internet;
- mBlock drivers for controlling Arduino installed on the computers.
- Circuit with all the devices needed to implement all the different codes, which needs the following components:
  - Arduino 1
  - Breadboard
  - Cables
  - 1k Ohms resistor
  - Button
  - 3x330 Ohms resistors
  - 3 LED lights
  - 1 Piezo Buzzer
  - 1 Ultrasound sensor

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

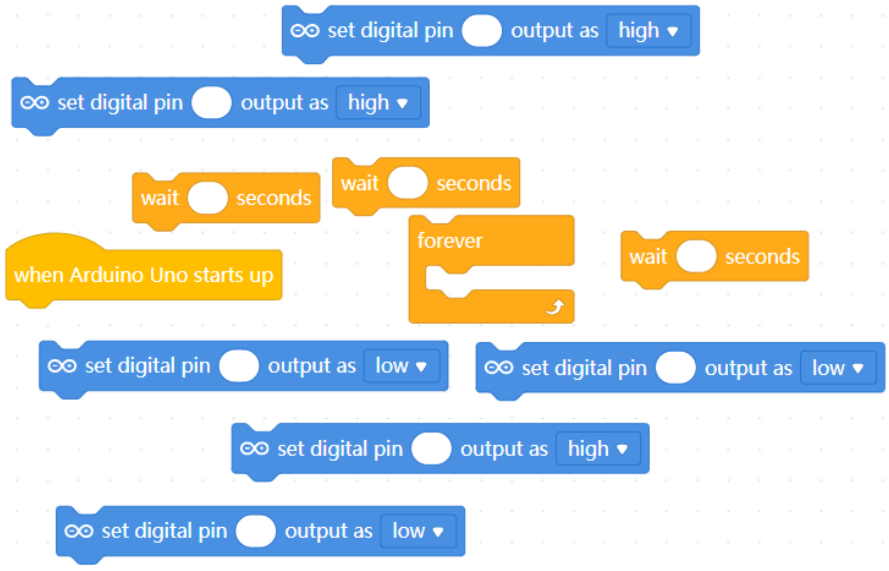


Description of the activity:

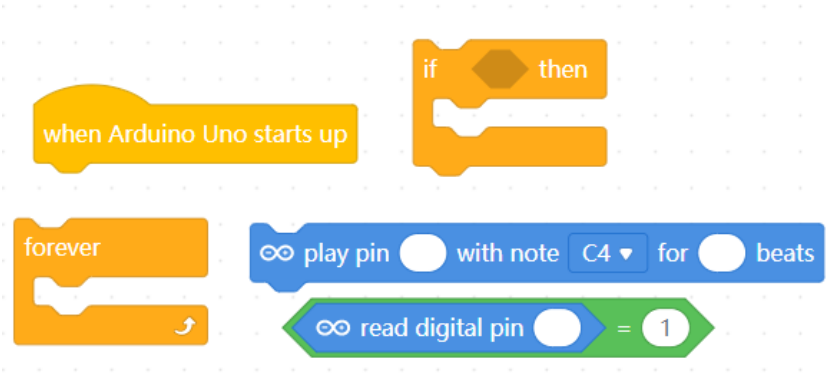
Challenge 1: Giving the circuit to the participants, the trainer will ask the participants to programme the circuit in order to make blink 1 single LED. If needed, the trainer can give the blocks that have to be used as a tip.



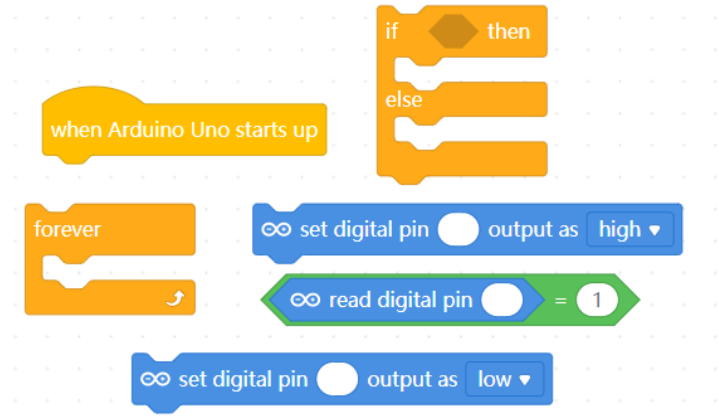
Challenge 2: The second activity will be to programme the 3 LED lights to behave as a traffic light.



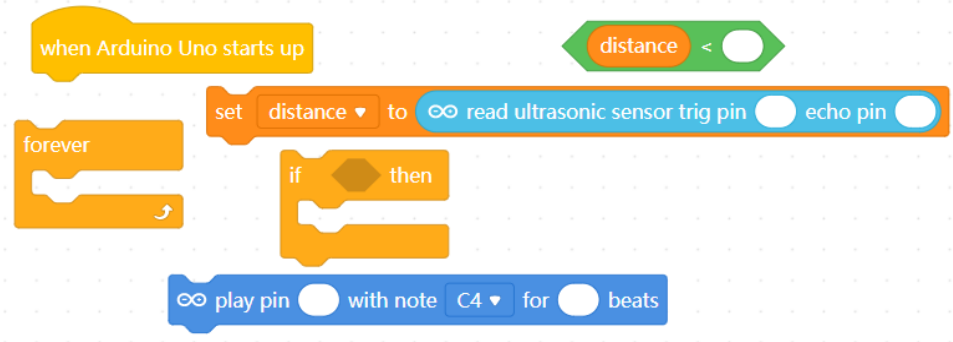
Challenge 3: The third activity consists of making the alarm go on (make the piezo buzzer produce a sound), only when the button is pushed. This activity aims to introduce the "if" structure.



Challenge 4: The fourth activity, intended to introduce the "if/else" block, is to turn on the light only when the button is pushed, otherwise it should be off.

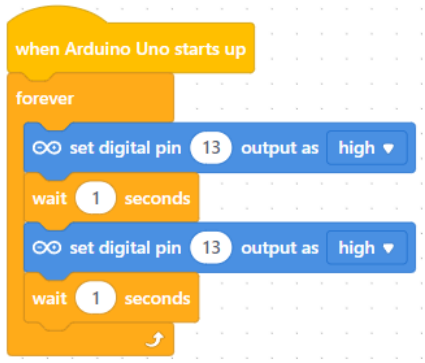


Challenge 5: The fifth and last activity, thought to introduce variables, is to read the ultrasonic sensor input signal and, if the object is too close, make the alarm go on.

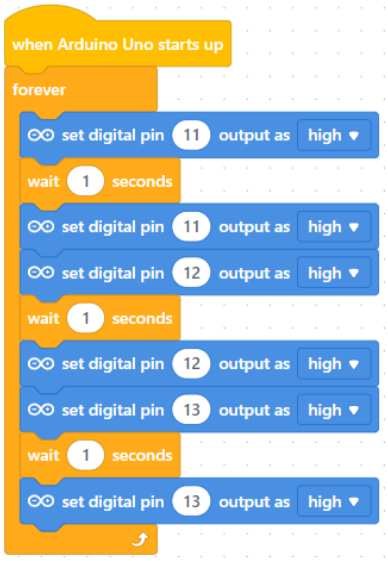


- **Solutions**

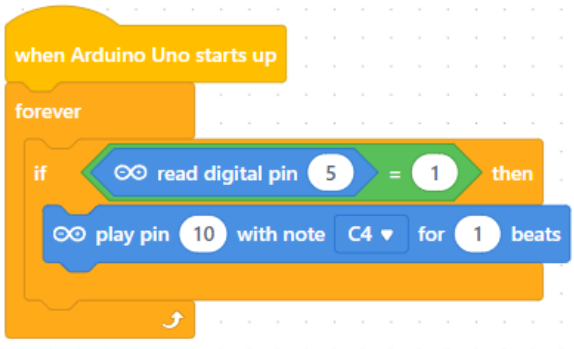
Challenge 1



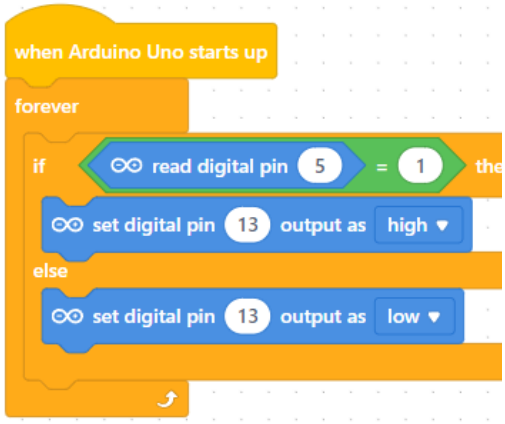
Challenge 2



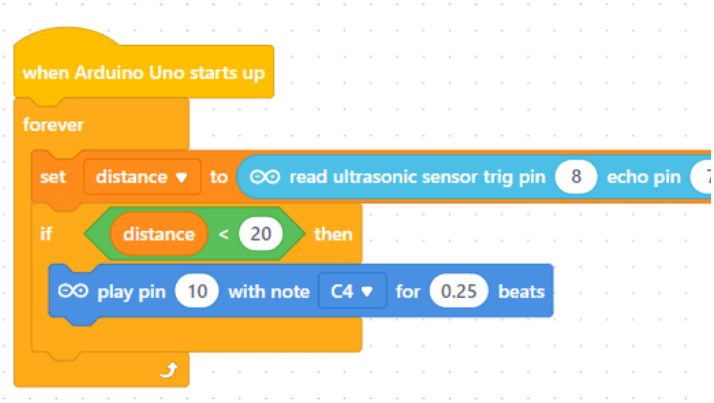
Challenge 3



Challenge 4



Challenge 5



How to adapt to different learners:

The trainer can encourage the participants to create the code from scratch by themselves. If they start to have problems the trainer can give them the blocks that have to be used in order to create the correct code.

If some participants have a high level of coding experience, they can skip the step of using block codes with mBlock and work directly with the Arduino IDE.

Additional information:

mBlock IDE: https://ide.mblock.cc/#/

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

Module: Microcontroller

Topic: Arduino Basics / Introduction into (Block)Coding
Task sheet A4.0: Learning if/else-query and while-loop

Time: 1:00 hour

General description:

To get into coding the participants need to understand some basic functions every programming language uses. With little and easy programmes, the participants get used to queries and loops before starting to use them in more complex ways.

Learning objective(s):

- Understand the structure and logic of the if/else-query
- Understand the structure and logic of the while loop
- Be able to implement simple commands

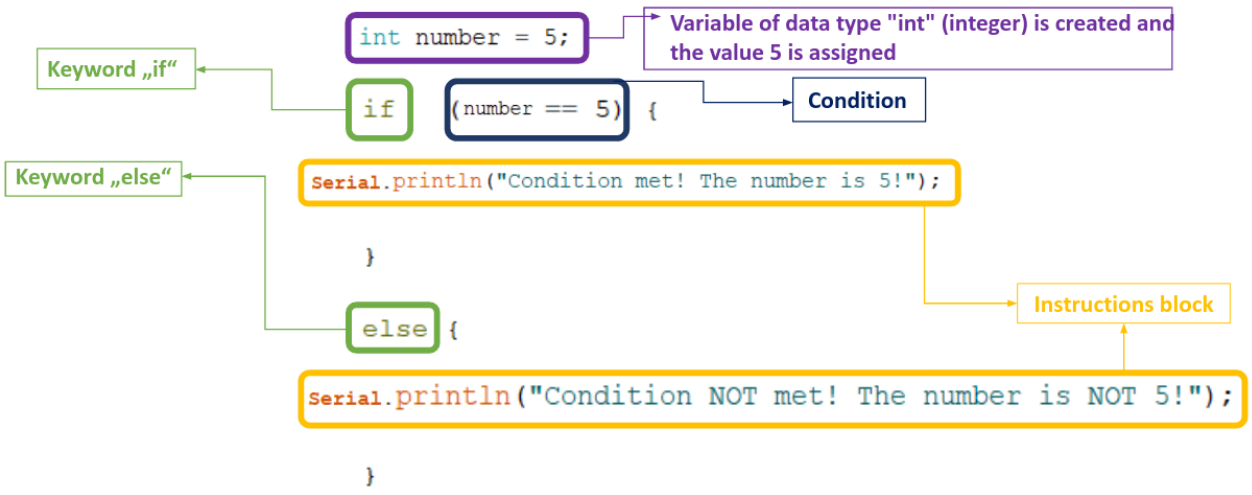
Material required:

- Microcontroller of your choice (Arduino Uno was used here)
- Laptop with programming software for the chosen microcontroller

Description of the activity:

If/else-query:

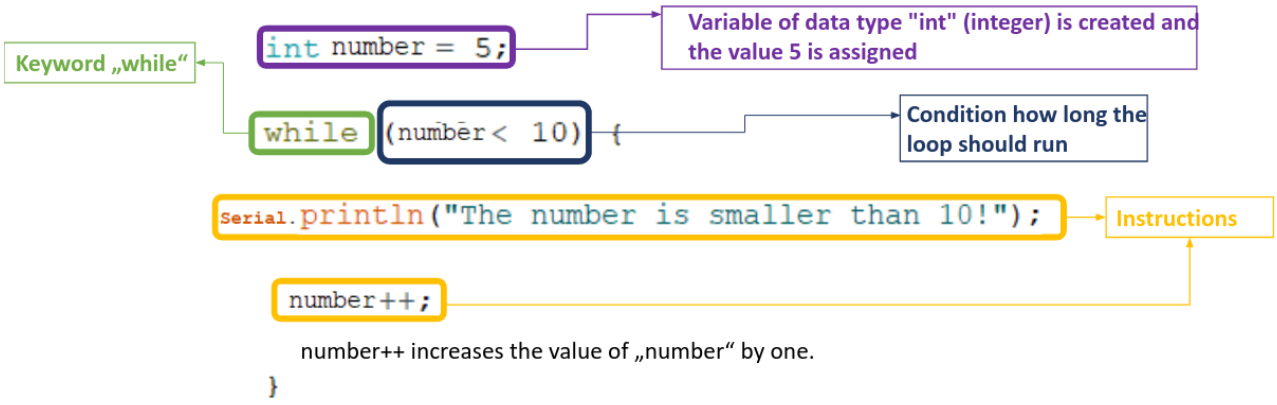
1. To teach the if/else-query you can (but do not have to) use the following picture:



2. After explaining you can let the participants experience the effects by turning the onboard LED on for a period of time if the condition is met and another period of time if the condition is not met. By using different periods of time the participants can clearly see which block (if or else) is being executed. Let the participants play around a bit and get familiar with this construct.

While-loop:

1. To teach the while-loop you can (but do not have to) use the following picture:



2. After explaining you can let the participants try it out and turn the LED on while the condition is met. When the condition is not met anymore the LED shall turn off. Do not use delays in the beginning so that the participants can see how fast the microcontrollers actually process instructions and loops.

How to adapt to different learners:

This task sheet is very basic. Still some participants may experience difficulties while trying to implement the if/else-query or the while-loop. To make sure participants can re-check the syntax, it might be better to print out the pictures above and give them an example.
Participants with more knowledge can try to define more complex conditions. For example, they can try to define a condition where the LED only turns on when the number is even and turns off when the number is uneven.

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

Module: Microcontrollers

Topic: Arduino Basics

Task sheet A4.1: Controlling a LED using a push button

Time: 2:00 hours

General description:

Pushing buttons or switches are often used to trigger an event/system on an electronic circuit. This activity will demonstrate how to connect a pushing button to your circuit, which will be used to control a LED. The assembly of this circuit will be done using the simulation software Tinkercad, therefore a registered account is needed.

Learning objective(s):

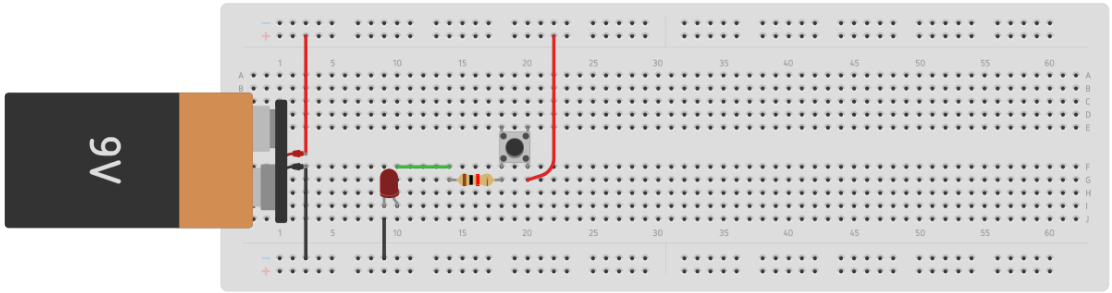
- Understand how to switch a LED ON/OFF using a button;
- Familiarise with triggers and events;
- Be able to connect different components in the breadboard.

Material required:

- Laptop with access to Internet;

Description of the activity:

1. Create a circuit project on Tinkercad;
2. Add the following material from the “components” section into your project:
    - 9V battery;
    - LED;
    - Resistor;
    - Pushbutton;
    - Breadboard;
    - Jumper wires
3. Connect the components to the breadboard according to the image below:



4. With the simulation ongoing, press the button in order to turn ON the LED.

How to adapt to different learners:

- The trainer can encourage participants to connect the components by themselves after all components have been added to the project. If they experience difficulties, they might follow the suggestion below:
  - Connect the 9v battery to the breadboard:
  - Connect the positive terminal of the battery to the positive pole on the board. Likewise, connect the negative terminal of the battery to the negative pole on the breadboard. You can change the colours of the wires/jumpers in the pop-up that will show up after each connection;
  - Connect a resistor to the board;
  - Connect the LED to the board and add a jumper to each of the LED pins;
  - Add a pushbutton to the board. Finally, connect the button pins with a jumper;
  - You can now simulate your project by clicking on the “start simulation” button on the upper-right section;

- To make it more exciting, if it is possible, have participants assemble the circuit physically so they can put into practice what they have learnt.

- Participants with more knowledge can be challenged to create other types of circuits, such as a circuit with multiple push buttons to switch ON different LEDs or use different components.

Additional information:

- Setting up a circuit with a push button and a LED:
  https://www.youtube.com/watch?v=ksNbEuhO4fU

- Circuits with multiple push buttons:
  https://www.youtube.com/watch?v=qY71RhHTFEo
  https://www.youtube.com/watch?v=Y23vMfynUJ0

- Timed LED with a button:
  https://www.youtube.com/watch?v=oPDcVnPfjdk

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

Module: Microcontrollers

Topic: Arduino Basics

Task sheet A4.2: Connecting a PIR sensor to an Arduino

Time: 2:00 hours

General description:

A sensor module is an electronic device that uses a sensor to detect nearby motion. Modules like this are used in many applications in real life (such as modern security alarm systems, automatic light switches, and automatic door and garage openers). Therefore, sensors like this are very useful to trigger some electronic operation when the presence of humans is detected.
This task will use a PIR sensor module that will turn on a LED when motion is detected.

Learning objective(s):

- Understand the concept behind PIR sensors and how they work;
- Code a PIR sensor;

Material required:

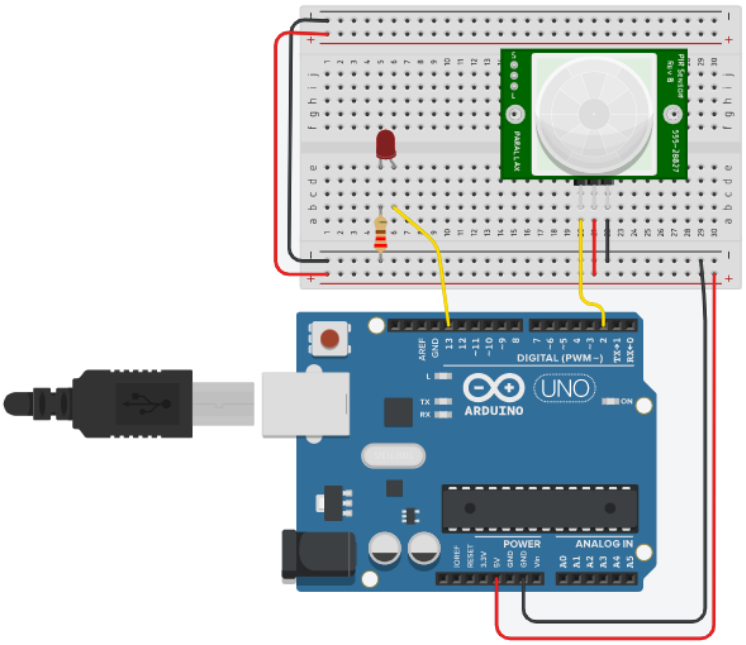
- Laptop with access to Internet

Description of the activity:

1. Create a circuit project on Tinkercad;

2. Add the following material from the “components” section into your project:
    - Sensor PIR
    - Arduino with cable
    - Jumpers
    - LED;
    - Resistor;
    - Breadboard;
    - Jumper wires

3. Connect the components to the breadboard according to the image below:



4. Add the following code:

```
int sensorState = 0;

void setup()
{
  pinMode(2, INPUT);
  pinMode(13, OUTPUT);
  Serial.begin(9600);

}

void loop()
{
  // read the state of the sensor/digital input
  sensorState = digitalRead(2);
  // check if sensor pin is HIGH. if it is, set the
  // LED on.
  if (sensorState == HIGH) {
    digitalWrite(13, HIGH);
    Serial.println("Sensor activated!");
  } else {
    digitalWrite(13, LOW);
  }
  delay(10); // Delay a little bit to improve simulation
performance
}
```

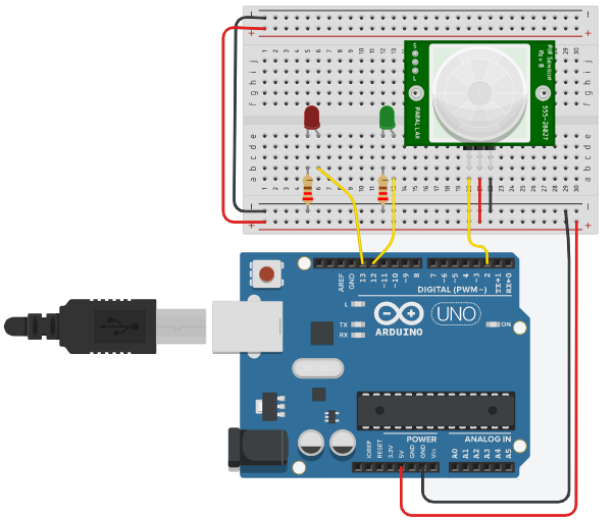
5. With the simulation ongoing, press the button in order to turn ON the LED.

How to adapt to different learners:

- The trainer can encourage participants to connect the components by themselves after all components have been added to the project. If they experience difficulties, they might follow the suggestion below:
  - Search for an Arduino board and connect it to the assay board by dragging it over the assay board. To rotate the battery (or any component), select it with the right mouse button and click on the “rotate” icon in the upper left section;
  - Connect a resistor to the board;
  - Connect the LED to the board and add a jumper to each of the LED pins;
  - Add a PIR sensor to the board. Finally, connect the button pins with a jumper;
  - Add the code provided.
  - Upload to the microcontroller
  - Click on the PIR sensor and observe the results.

- To make it more exciting, if it is possible, have participants assemble the circuit physically so they can put to practice what they have learnt.

- Participants with more knowledge can be challenged to create other types of circuits, such as circuits with more LEDs. For instance:
  - Green LED turns on whenever there is no movement, thus showing that the system is active;
  - Red LED turns on whenever there is movement and the green LED turns off, starting at the beginning.



Additional information:

- **How PIR Sensor Works and how to use It with Arduino**
  https://www.youtube.com/watch?v=6Fdrr_1guok

- **Connecting a PIR sensor to an Arduino**
  https://www.youtube.com/watch?v=FxaTDvs34mM

- **PIR Sensor Arduino Project – Motion Detector**
  https://www.youtube.com/watch?v=vmhPQb4rdPw

- **Information about the PIR sensor:**
  https://www.mouser.com/datasheet/2/737/pir-passive-infrared-proximity-motion-sensor-932858.pdf

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

Module: Microcontrollers

Topic: Arduino Basics

Task sheet A4.3: Connecting an ultrasonic sensor to Arduino

Time: 2:00 hours

General description:

An ultrasonic sensor module is one of the most common methods used to detect the existence of objects or obstacles in front of robots, as well as to measure the distance from an object. In order to do this, the sensor sends ultrasound signals and monitors its reflection. In this activity, the ultrasonic sensor SRF 05 will be used. Participants will start by connecting the sensor to the Arduino board, then code it and control the distance in the serial monitor using, e.g., hand proximity.

Learning objective(s):

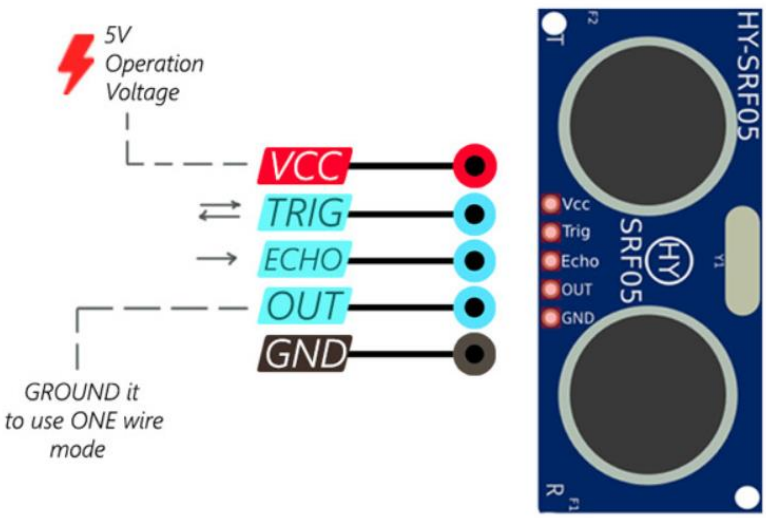
- Understand the concept behind ultrasonic sensors and how they work;
- Use basic coding an ultrasonic sensor;

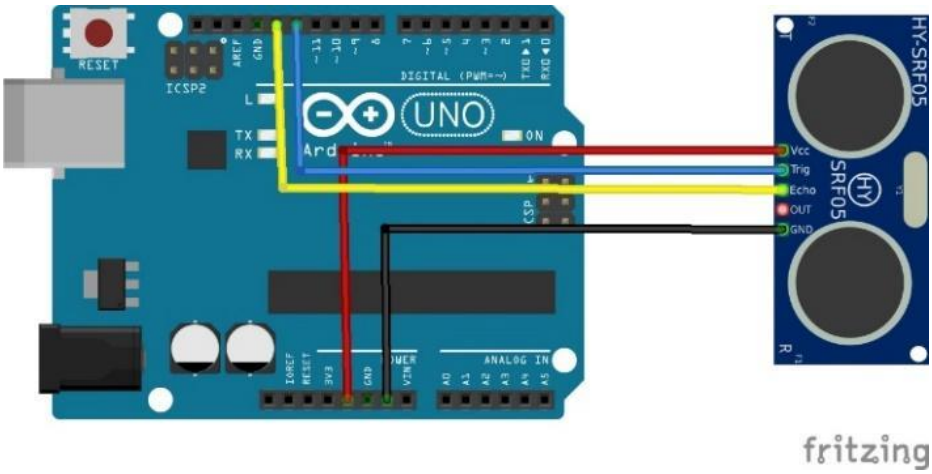
Material required:

- SRF05
- Arduino with cable
- Jumpers

Description of the activity:

1. Connect a SRF 05 ultrasonic sensor to an Arduino board.





2. Add the following code to the Arduino to control the monitor:

```
/*
VCC to +5V
GND to ground
TRIG to digital pin 12  //You can use other digital pin
ECHO to digital pin 13  //You can use other digital pin
*/

const int TRIG_PIN = 12;
const int ECHO_PIN = 13;

void setup()
{
  // initialize serial communication:
  Serial.begin(9600);
  pinMode(TRIG_PIN,OUTPUT);
  pinMode(ECHO_PIN,INPUT);
}


void loop()
{
  long duration, distanceCm, distanceIn;

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN,HIGH);

  // convert the time into a distance
  distanceCm = duration / 29.4 / 2 ;
  distanceIn = duration / 74 / 2;

  if (distanceCm <= 0)
  {
    Serial.println("Out of range");
```

```
  }
  else
  {
    Serial.print(distanceIn);
    Serial.print("in: ");
    Serial.print(distanceCm);
    Serial.print("cm");
    Serial.println();
  }
  delay(1000);
}
```

3. Upload to the microcontroller and look at the values in the serial monitor.

How to adapt to different learners:

- There are other types of sensors that can be used to measure distance, namely the Sharp IR Distance Sensor, which can not only detect objects but also measure its distance.

Figure 1- Sharp Sensor

This Long Range IR Distance Sensor is a great alternative to the ultrasonic sensor and infrared sensors that cannot reach an equal or greater range. Its main application is in the detection of objects that are at a distance of 10 to 80cm, which is used in different applications, especially in the automatic parking system (the sound gets higher as it gets closer to the object).
Have participants try out this sensor if it is available.

Additional information:

- **Tutorial videos on sensors:**

  Connect a ultrasonic sensor to an Arduino with different sensors:
  https://www.youtube.com/watch?v=GL8dkw1NbMc
  https://www.youtube.com/watch?v=Lx7KEaRLOU0
  https://www.youtube.com/watch?v=wCOPLVgNpcY
  https://www.youtube.com/watch?v=GL8dkw1NbMc

  Interfacing ultrasonic sensor with Arduino (using Tinkercad)
  https://www.youtube.com/watch?v=Om3L02X6_KU

- **Information about the sensors used**:

  Ultrasonic sensor SRF 05:
  https://picaxe.com/docs/srf005.pdf
  https://www.robot-electronics.co.uk/htm/srf05tech.htm

  Sharp distance sensor:
  https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y0a21yk_e.pdf

- **Information about the speed of sound that can be used to change values in the original code of the task:**

The speed of sound is 340 m / s, or 0.034 cm / µs (that is, approximately 29.4 µs per centimeter covered). Thus, sound waves can travel for about 588 µs between sensor and obstacle, according to the formula:

t = s/v

Where **t** is the time, **s** is the space covered by the ultrasound wave, and **v** is its speed of propagation in the air.

Supposing an obstacle is located 20 cm away from the sensor:

t = 20cm / 0,034cm/µs = 588,23 µs

In practice, the total travel time of the pulse will be twice that value because it is a round trip. In our example, the total time between the transmission of the ultrasound pulse and receiving it back is: **588,23 µs x 2 = 1176,46 µs**

Thus, to obtain the real distance to the object, we must divide the measured value by 2. So, the formula for determining the distance to the object will be:

s = t x 0,034 / 2 or **s = microseconds / 29,4 / 2**

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

Module: Microcontrollers

Topic: Arduino Basics

Task sheet A4.4: Creating an air conditioning system from scratch

Time: 3:00 hours

General description:

When creating electronic circuits, one of the typical behaviors is reading a specific signal from a sensor in order to control one or more actuators. In this activity, the participant will go through the whole process of creating a circuit where a LED and a motor will be controlled through the reading of a temperature sensor and a distance sensor. In this way, the air conditioning system will only be ON when there is a person present and the temperature goes above a certain threshold. This activity is thought to be made using the developing environment mBlock in order to help the participants with low coding skills get into the coding language, but if the participants are skilled enough, they can work directly with the Arduino IDE.

Learning objective(s):

- Understand how to read a sensor value and use a threshold to activate diverse other components;
- Familiarize with triggers and events;
- Be able to connect different components in the breadboard;
- Be able to expand a circuit and a code by adding new components

Material required:

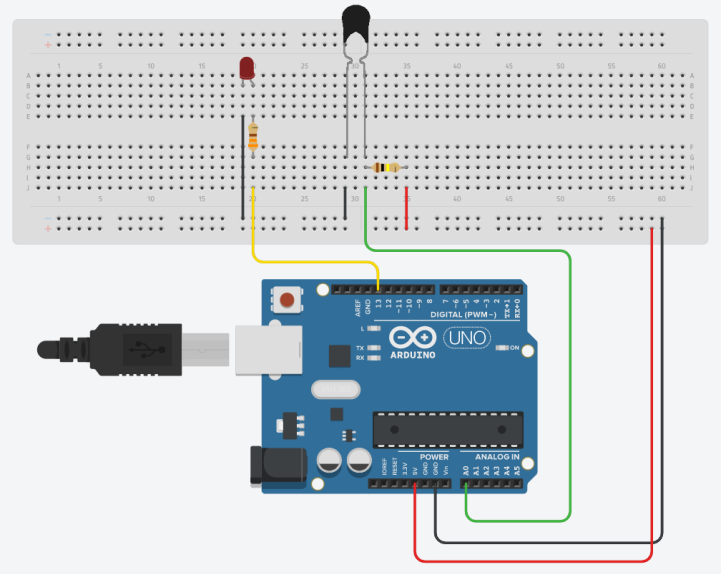
- Laptop with access to Internet;
- mBlock drivers for controlling Arduino installed on the computers.
- Arduino Kit or separate components: an Arduino (uno), breadboard, cables, resistors (220Ω & 100KΩ), 1 led, ultrasonic sensor or PIR sensor, DC motor with van, thermistor

Description of the activity:

1. The trainer will briefly explain how to use the mBlock developing environment in order to create the block codes that will control the components of the circuit.

2. Then, the trainer will ask to create a circuit where there will be a temperature sensor/thermistor and a LED light. The participants will have to turn on the LED

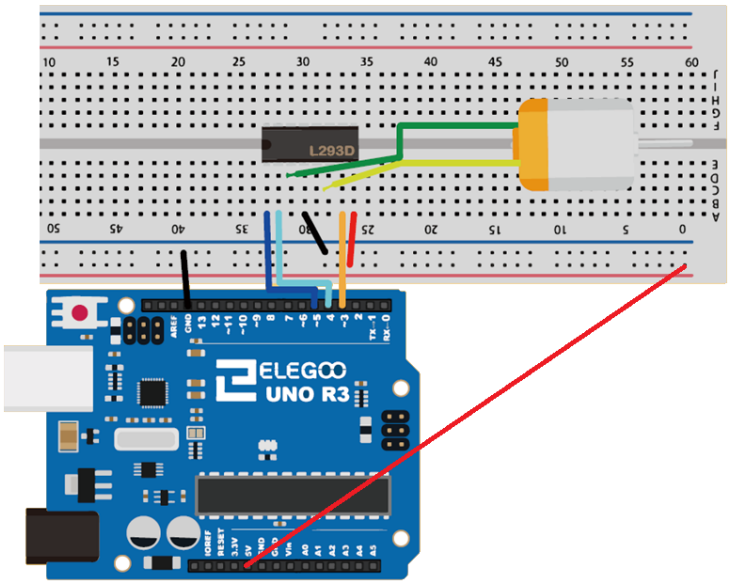
light if the temperature goes above 27 Celsius degrees. An example of circuit using a thermistor could be as follows:



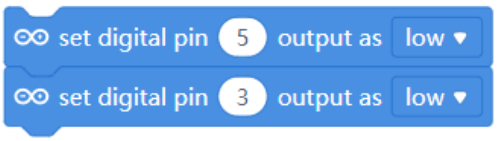
Take into account that if a thermistor is needed, the temperature value should be calculated following the [Steinhart-Hart equation](). The application of the formula to the thermistor value can be done as follows:

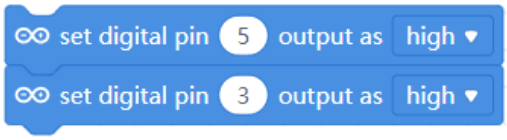

```
Vo = analogRead(ThermistorPin);
R2 = R1 * (1023.0 / (float)Vo - 1.0);
logR2 = log(R2);
T = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2));
Tc = T - 273.15;
```

3. After doing this circuit correctly, the participants will add a motor with a fan in order to start simulating the air conditioning system. The circuit can be as follows:

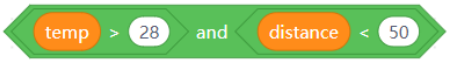


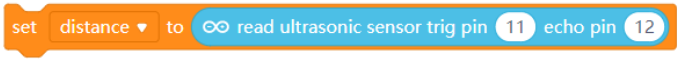
Depending on the level of the participants the new block codes needed can be shown as a tip:





4. The next step is adding an ultrasound sensor or a PIR sensor, so that when a presence is detected, the air conditioning system will be turned ON. In this way it will only work when a person is present in the room. As in the previous step, depending on the level some tips can be given to the participants:





5. The last step is to make the participants use the Arduino IDE and understand how to use the serial monitor in order to monitor the value of the sensors (distance and temperature). They will have to download the code from mBlock and add it to Arduino in order to print the value of the sensors on each cycle of the loop. If the participants have done this exercise without mBlock but using Arduino IDE, this step can be done together with the previous steps.

How to adapt to different learners:

- The trainer can encourage participants to connect the components by themselves, building always on the previous circuit to arrive at the final one. They can use the internet for help. If they experience difficulties, the trainer might give them the images with the diagram of the circuit.
- The same way, the trainer can encourage the participants to create the code from scratch by themselves. If they start to have problems the trainer can give them the blocks that have to be used in order to create the correct code.
- If some participants have a high level of coding experience, they can skip the step of using block codes with mBlock and work directly with the Arduino IDE. If done this way can be interesting for the participants to be guided in the process of presenting the problem and trying to arrive at the solution by themselves.
- Participants with more knowledge can be pushed to try to improve this circuit in other ways, for example a button that will turn on the system without checking the sensors, or a guided fan to where the presence in the room is detected by the sensors.

Additional information:

Steinhart-hart equation:

https://en.wikipedia.org/wiki/Steinhart%E2%80%93Hart_equation

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

Module: Microcontroller

Topic: IoT

Task sheet A5.1: Arduino weather station

Time: 2:30 hours

General description:

Using the multiple sensors with an Arduino board a weather station will be created. The weather station can gather data for temperature, pressure, light intensity and upload them online (using a service such as thingspeak.com).

Learning objective(s):

- Familiarise with the use of different sensors (DHT11, BMP180)
- Learn to gather data and upload them on a channel (thingspeak.com)
- Familiarise with the use of LDR (Light Dependent Resistor)

Material required:

- Adafruit BMP180
- DHT11 Temperature & Humidity Sensor (4 pins)
- LDR (Light Dependent Resistor)
- Espressif ESP8266 ESP-01
- Arduino Nano R3
- DC jack
- switch
- 12v-2A wall adapter
- PCB
- Male Header 40 Position 1 Row (0.1")
- Male Header 40 Position 1 Row (0.1") (x5)

Necessary software:

- Arduino IDE
- ThingSpeak API

Other Required materials:

- wire stripper
- Hot glue gun (generic)
- Soldering iron (generic)
- screw driver
- plastic box
- File (used for handcrafting)

Description of the activity:

1. The trainer will do an introduction to Arduino and the different sensors that can be used with it and the way the gathered data can be uploaded online and how they can be used.
2. Depending on the level of the participants, the trainer will ask them to try to programme the microcontroller with the help of online sources or will provide them with the code, asking them to try to understand it.
3. The trainer will ask the participants to build the circuits and connect the sensors, providing them some diagrams.
4. Once the circuit is built and programmed, the trainer will ask the participants to create an account to a platform where they can upload the data in their own channel.
5. When everything is connected and working, the participants will test the weather station and the trainer will help them if there are any problems (depending on the level of participants and the errors that came up, the trainer may offer guidance to solve any programming errors or check the circuits).

How to adapt to different learners:

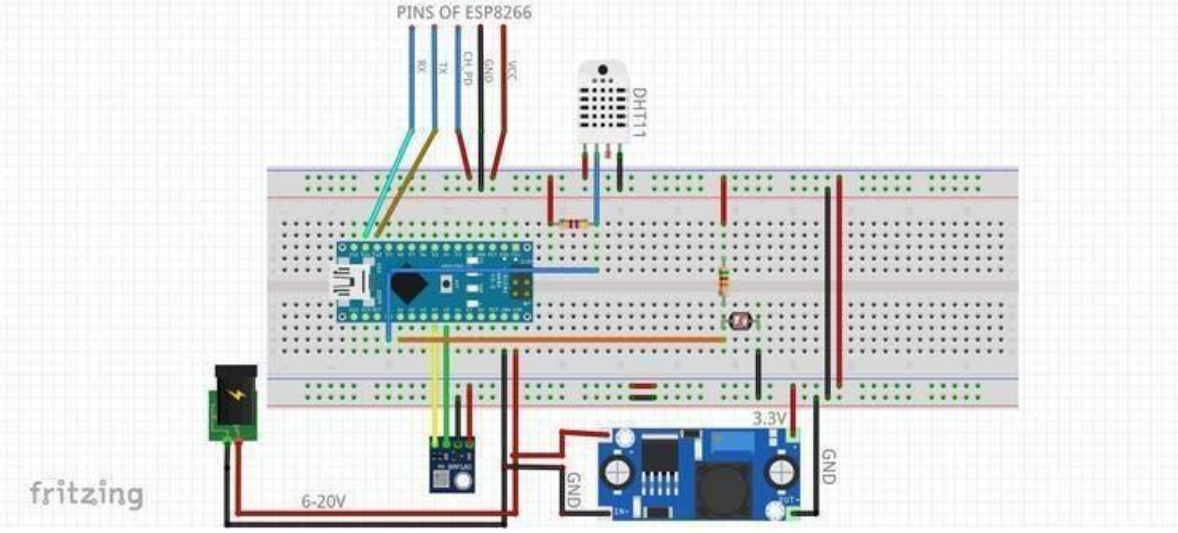
- The trainer can provide the code or ask the participants to look for the appropriate code online or guide them to make the needed modification on the code they will find.
- For participants in the beginner level the trainer may explain the code or give the code divided in snippets and allow the participants to use it in the correct order.

Additional information:

- How to build the circuit:

  Connections:

  - BMP180 connects to I2C port of arduino nano.
  - LDR is connected in voltage divider fashion with 51 KOhm resistor and junction is connects to A1 pin of arduino nano.
  - DHT11's data pin is pulled high with 4.7 KOhm resistor and connected to A0 pin of arduino nano.
  - ESP8266's TX and RX connects to D10 and D11 of arduino nano respectively. ESP8266's CH_PD connects to 3.3V rail.
  - Adjust LM2596 module's output to 3.3V by turning potentiometer on this module. Connect output of this module to Vcc and Gnd of BMP180,DHT11,LDR and ESP8266's Vcc and Gnd respectively.
  - Input of LM2596 module comes from 12V-2A wall adapter which also connects to Vin and Gnd of Arduino nano.



Assembling circuit on General Purpose Circuit board (GCB)
Hardware tools and extra items from step 3 are now in use.

- Use female berg strip for Arduino nano and ESP8288's placement on GCB,
- Use solder iron and solder wire to connect them electrically to the board (to ensure that they are not accidentally removed while you are working),
- Use female connectors to extend the reach of all sensors and LM2596 module as they will be stuck to the lid and wall of enclosure,
- Use male berg strip to make connecting points for female extensions made above,
- Realize circuit schematic on GCB using wires (strip them using wire stripper), or rail of melted solder wire and finally,
- Check for sorts before powering the circuit using a multimeter

Code:
https://github.com/jayraj4021/Personal-Weather-Station-14

Before you burn the code take care of following things:

- Make sure that all libraries are installed,
  - #include "DHT.h"
  - #include <LiquidCrystal.h>
  - #include <SFE_BMP180.h>
  - #include <Wire.h>
- Replace hyphens with SSID of your access point (wifi router) in line 14 of the code,
- Replace hyphens with PASSWORD of your wifi network in line 15 of the code,
- Replace hyphens with your ThingSpeak's private channel write API key in line 17 and
- While programming Arduino nano make sure that your 12V DC supply is OFF.

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

Module: Microcontroller

Topic: Internet of Things

Task sheet A5.2: Create your local server

Time: 2:00 hours

General description:

The students will learn how to use the wi-fi microcontroller (for example, ESP32 or ESP 8266), in order to create their own webserver and be able to control a pair of LEDs through a web interface. This task sheet allows combining Arduino circuits with web development languages. The activity includes quite complex code and can be adapted from an intermediate level (with some knowledge of HTML and Arduino coding language) to an advanced level.

Learning objective(s):

- Understand what is the Internet of Things;
- Learn how to set up the microcontroller to works as a web server;
- Learn how to create a webpage inside the microcontroller that will be offered when the client accesses the server;
- Learn how to link the actions on the web page to particular exits in order to turn on and off some LEDs.

Material required:

- Computer or laptop
- Internet connection
- Text editor (online or offline): Sublime Text/Brackets/W3Schools online editor
- Wi-Fi Microcontroller, for example, ESP32 or ESP8266.
- USB cable to connect the microcontroller to the computer
- 1 Breadboard
- 3 LEDs lights
- 3 Resistors (220 or 330 ohms)
- Jumper wires

Description of the activity:

1. The trainer will do an introduction on the wi-fi microcontroller. Wi-fi microcontrollers like the ESP32 and ESP8266 are a kind of microcontroller that can send and receive information through a Wi-Fi connection. In this activity we will use this microcontroller to create a web server.
2. As a trainer, you can follow the instructions described in this tutorial for the installation of the wi-fi microcontroller:

https://randomnerdtutorials.com/esp8266-web-server/

3. After the board installation the students can start to programme the microcontroller. Depending on the level of the participants, the trainer will ask them to try to programme the microcontroller with the help of online sources or will provide them with the code, asking them to try to understand it.
4. The trainer will ask the students to build a LED circuit with only 1 led and make a test of the activity, using the web interface to control the state of the LEDs.
5. Once the participants have managed to make the LED circuit work, the trainer can ask to modify the circuit and the code to work with 2 LED lights instead of 1 and/or to change the web page delivered by the server, changing style, text, elements, etc.

How to adapt to different learners:

- The trainer can provide the code or ask them to look for the code online for themselves depending on how resourceful they are in looking for information on the internet.
- Advanced students can also try to add more different kinds of sensors and actuators in the circuit and try to control them with the web interface.

Additional information:

- HTML reference guide
- W3Schools - Guide for every HTML element and CSS rule, and examples for each one of them
- Code Example 1
- Code Example 2:

```
#include <ESP8266WiFi.h>
  #include <WiFiClient.h>

  //ESP Web Server Library to host a web page
  #include <ESP8266WebServer.h>

  //------------------------------------------------------------  //Our  HTML
  webpage contents in program memory
  const char MAIN_page[] PROGMEM = R"=====(
  <!DOCTYPE html>
  <html>
  <body>
  <center>
  <h1>WiFi LED on off demo:</h1><br>
  Click to turn <a href="led1On" target="myIframe1">LED 1 ON</a><br>
  Click to turn <a href="led1Off" target="myIframe1">LED 1 OFF</a><br>
  LED State:<iframe name="myIframe1" width="150" height="50"
    frameBorder="0"></iframe><br>
  <hr>
  </center>
```

```
</body>
</html>
)=====";
//---------------------------------------------------------------   //On   board   LED
Connected to GPIO2
#define LED1 2

//SSID and Password of your WiFi router
const char* ssid = "WifiName";
const char* password = "password";

//Declare a global object variable from the ESP8266WebServer class.
ESP8266WebServer server(80); //Server on port 80

//===============================================
// This routine is executed when you open its IP in browser
//===============================================
void handleRoot() {
  Serial.println("You called root page");
  String s = MAIN_page; //Read HTML contents
  server.send(200, "text/html", s); //Send web page
}

void handleLED1on() {
  Serial.println("LED 1 on page");
  digitalWrite(LED1,HIGH); //LED 1 on
  server.send(200, "text/html", "ON"); //Send ADC value only to client  ajax
    request
}

void handleLED1off() {
  Serial.println("LED 1 off page");
  digitalWrite(LED1,LOW); //LED 1 off
  server.send(200, "text/html", "OFF"); //Send ADC value only to client  ajax
    request
}
//===============================================
// SETUP
//===============================================
void setup(void){
Serial.begin(115200);
Serial.println("");
WiFi.mode(WIFI_AP); //We use the ESP8266 only as access point
WiFi.softAP(ssid, password); //Start HOTspot
```

```
IPAddress myIP = WiFi.softAPIP(); //Get IP Address
Serial.print("HotSpt IP:");
Serial.println(myIP);

//Onboard LED port Direction output
pinMode(LED1,OUTPUT);
//Power on LED state off
digitalWrite(LED1,HIGH);

server.on("/", handleRoot); //Which routine to handle at root  location. This is
    display page
server.on("/led1On",  handleLED1on);  //as  Per  <a  href="led1On">,
    Subroutine to be called
server.on("/led1Off", handleLED1off);

server.begin(); //Start server
Serial.println("HTTP server started");
}
//=====================================================
// LOOP
//=====================================================
void loop(void){
server.handleClient(); //Handle client requests
}
```

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

Module: Microcontroller

Topic: Introduction to IoT

Task sheet A5.3: Sensing the Environment & Notifying

Time: 01:30 hour(s)

General description:

In this exercise, we are using a Python programme to allow our Raspberry Pi to first sense the environment around it using HAT, printing temperature on the LED matrix and then, based on certain triggers it would send notifications using InstaPush API on your android phone.

Learning objective(s):

- Familiarise with the use of sensors with Raspberry Pi
- Learn to gather data and print them on an output (LED matrix)
- Familiarise with push notifications

Material required:

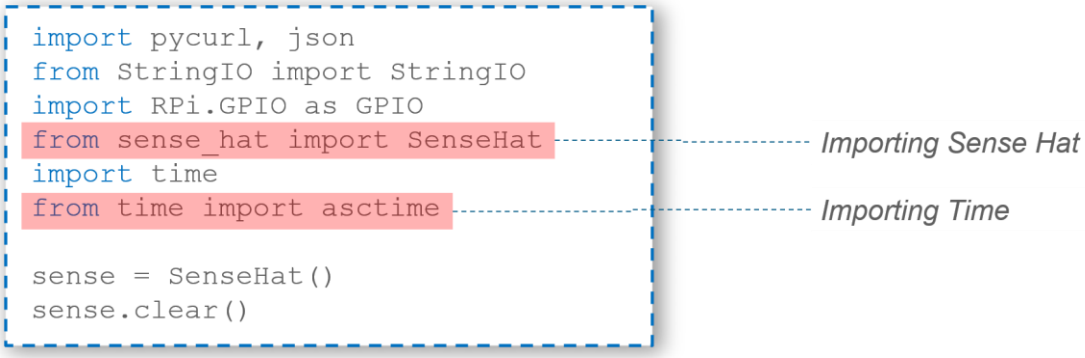
- Raspberry Pi
- LED matrix
- InstaPush ID (or alternative such as: Pusher Beams)
- NOOBS software
- Notepad++ (software)

Description of the activity:

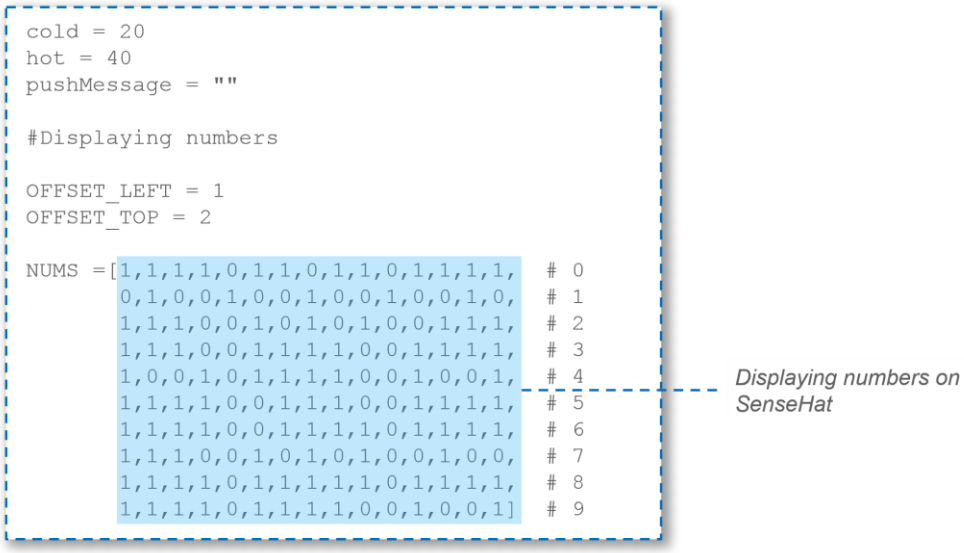
- Create an InstaPush ID and then an application (http://instapush.im/ - https://dashboard.pusher.com/accounts/sign_up)
- Proceed by creating the events, trackers and the push message
- Download the Instapush application
- Note down the Application ID and Application Secret
- Import the required packages

For the Python programme (the code can be found at the end of the task sheet):

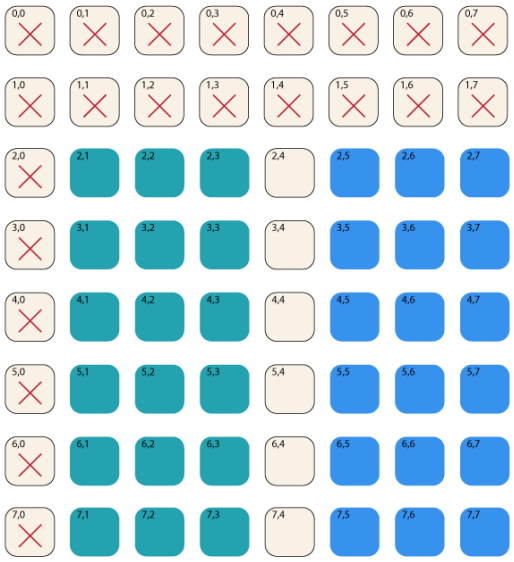
First of all you need to import packages as shown in the below image:



Importing json module allows you to encode python objects as JSON strings, and decode JSON strings into python objects. We will be using this for sending push messages. Then the next one is pycurl, which is a Python interface to libcurl, the multiprotocol file transfer library. Similar to the urllib Python module, pycurl can be used to fetch objects identified by a URL from a Python program. Rpi.GPIO is for basic Raspberry Pi input-output operations and using Sense_hat we can control our HAT sensors and display the data over the LED matrix.



Now here we have defined the hot and cold temperature after which the event will be triggered and notification will be sent. Led matrix has 8*8 i.e. 64-pixel positions. OFFSET form left and the top is defined to leave the number columns & rows simultaneously from top & left as shown in the figure below.



IOT Tutorial: HAT LED Matrix

The NUMS matrix contains 3*5 i.e. 15 pixels, i.e. 3 columns and 5 rows. The numbers tell which pixel should be on & which pixel should be off. We will display 2 digits, one digit will be displayed in the blue region as shown in the above figure and tens digit will be displayed in the green region. Now let us know how to write the logic for displaying a single as well as a two-digit number.

```
# Displaying a single digit (0-9)
def show_digit(val, xd, yd, r, g, b):
  offset = val * 15
  for p in range(offset, offset + 15):
    xt = p % 3
    yt = (p-offset) // 3
    sense.set_pixel(xt+xd, yt+yd, r*NUMS[p], g*NUMS[p], b*NUMS[p])

# Displays a two-digit positive number (0-99)
def show_number(val, r, g, b):
  abs_val = abs(val)
  tens = abs_val // 10
  units = abs_val % 10
  if (abs_val > 9):
     show_digit(tens, OFFSET_LEFT, OFFSET_TOP, r, g, b)
  show_digit(units, OFFSET_LEFT+4, OFFSET_TOP, r, g, b)
```

Displaying numbers on SenseHat

The first function explains how to display a digit. It accepts arguments as the number, offset added in the default offset to their x and y-axis, and RGB code in which the pixel needs to be displayed. Then the offset variable in the show_digit function declared here, is to skip the pixels of the above number and only pick 15 pixels of the corresponding number from the NUMS matrix. xt is modulo 3 as we have only three pixels in each row. At the last set_pixel function will display pixels on the LED matrix.
Now for displaying two-digit numbers, first we are separating tens and ones digit and then calling show_digit function both tens digit and ones digit. We are adding 4 columns for displaying one digit to separate ones and tens digit.

```
appID = "59bb6e6ba4c48a1cd674e33d" #Adding Instapush Details

appSecret = "fd127d824390296b5f84818cddafeebe"# Add your Instapush Secret Key
pushEvent = "TempNotify"
c = pycurl.Curl()
c.setopt(c.URL, 'https://api.instapush.im/v1/post') # Setting API URL
c.setopt(c.HTTPHEADER, ['x-instapush-appid: ' + appID,
                        'x-instapush-appsecret: ' + appSecret,
                        'Content-Type: application/json'])

buffer = StringIO() # Capture response from push API call
```

Validating Instapush

Then, in our next step, we are creating a pycurl instance and passing all the required details to connect to the InstaPush API like app ID, app Secret in the header.

```
def pushmessage():

      json_fields = {}

      json_fields['event']=pushEvent
      json_fields['trackers'] = {}
      json_fields['trackers']['message']=pushMessage
      #print(json_fields)
      postfields = json.dumps(json_fields)

      c.setopt(c.POSTFIELDS, postfields)
      c.setopt(c.WRITEFUNCTION, buffer.write)
      c.setopt(c.VERBOSE, True)
```

Defining notification message

Now, we are specifying data which will be transmitted over the InstaPush API like the name of the push Event & trackers message. Afterwards, we are dumping the data. VERBOSE allows us to see the status of a transmitted message.

```
while True:

      temp = round(sense.get_temperature())
      humidity = round(sense.get_humidity())
      pressure = round(sense.get_pressure())
      message = ' T=%dC, H=%d, P=%d ' %(temp,humidity,pressure)
```

As we are all set with our InstaPush API for sending notification & logic for displaying numbers on the LED matrix, this is high time to sense the environment using HAT sensors for temperature, humidity and pressure. From the above image, you can see the functions used to get each of them respectively. round() function rounds off the obtained value to a whole number. At last, the message contains all the details regarding the environment which needs to be transmitted.

```
        if temp >= hot:
                pushMessage = "It is hot:  " + message
                pushmessage()
                c.perform()
                # Capture the response from the server
                body = buffer.getvalue()
                pushMessage = ""

        elif temp <= cold:
                pushMessage = "It is cold:  " + message
                c.perform()
                # Capture the response from the server
                body = buffer.getvalue()

        # Resetting the buffer
        buffer.truncate(0)
        buffer.seek(0)

c.close()
GPIO.cleanup()
```

As we were sensing the environment continuously, so here we are specifying the conditions which will trigger the event and send notification over the android phone.

After executing the programme, if the temperature drops or rises you will receive notifications as shown in the below figure.

How to adapt to different learners:

If the participants are struggling with the code, the trainer can provide them with the code and briefly explain it to them prior to running it.

Additional information:

- **Using Puser Beams for notification pushing to android:**
  https://pusher.com/tutorials/push-notifications-android

- **Code:**

```
import pycurl, json
from StringIO import StringIO
import RPi.GPIO as GPIO
from sense_hat import SenseHat
import time
from time import asctime

sense = SenseHat()
sense.clear()

cold = 37
hot = 40
pushMessage = ""

#####################################################
# Code for displaying number on LED Matrix

OFFSET_LEFT = 1
OFFSET_TOP = 2

NUMS =[1,1,1,1,0,1,1,0,1,1,0,1,1,1,1, # 0
0,1,0,0,1,0,0,1,0,0,1,0,0,1,0, # 1
1,1,1,0,0,1,0,1,0,1,0,0,1,1,1, # 2
1,1,1,0,0,1,1,1,1,0,0,1,1,1,1, # 3
1,0,0,1,0,1,1,1,1,0,0,1,0,0,1, # 4
1,1,1,1,0,0,1,1,1,0,0,1,1,1,1, # 5
1,1,1,1,0,0,1,1,1,1,0,1,1,1,1, # 6
1,1,1,0,0,1,0,1,0,1,0,0,1,0,0, # 7
1,1,1,1,0,1,1,1,1,1,0,1,1,1,1, # 8
1,1,1,1,0,1,1,1,1,0,0,1,0,0,1] # 9

# Displays a single digit (0-9)
def show_digit(val, xd, yd, r, g, b):
offset = val * 15
for p in range(offset, offset + 15):
xt = p % 3
yt = (p-offset) // 3
sense.set_pixel(xt+xd, yt+yd, r*NUMS[p], g*NUMS[p], b*NUMS[p])
```

```
# Displays a two-digits positive number (0-99)
def show_number(val, r, g, b):
abs_val = abs(val)
tens = abs_val // 10
units = abs_val % 10
if (abs_val > 9): show_digit(tens, OFFSET_LEFT, OFFSET_TOP, r, g, b)
show_digit(units, OFFSET_LEFT+4, OFFSET_TOP, r, g, b)

#######################################################################

temp = round(sense.get_temperature())
humidity = round(sense.get_humidity())
pressure = round(sense.get_pressure())
message = " T=%dC, H=%d, P=%d " %(temp,humidity,pressure)

#setup InstaPush variables
# add your Instapush Application ID
appID = "59bb6e6ba4c48a1cd674e33d"

# add your Instapush Application Secret
appSecret = "fd127d824390296b5f84818cddafeebe"
pushEvent = "TempNotify"

# use Curl to post to the Instapush API
c = pycurl.Curl()

# set API URL
c.setopt(c.URL, 'https://api.instapush.im/v1/post')

#setup custom headers for authentication variables and content type
c.setopt(c.HTTPHEADER, ['x-instapush-appid: ' + appID,
'x-instapush-appsecret: ' + appSecret,
'Content-Type: application/json'])

# use this to capture the response from our push API call
buffer = StringIO()
# # # # # # # # # # # # # # # # # # # # # # # # # # #

def p(pushMessage):
# create a dict structure for the JSON data to post
json_fields = {}

# setup JSON values
json_fields['event']=pushEvent
json_fields['trackers'] = {}
json_fields['trackers']['message']=pushMessage
#print(json_fields)
postfields = json.dumps(json_fields)

# make sure to send the JSON with post
c.setopt(c.POSTFIELDS, postfields)
```

```
# set this so we can capture the resposne in our buffer
c.setopt(c.WRITEFUNCTION, buffer.write)

# see the post sent
c.setopt(c.VERBOSE, True)

# setup an indefinite loop that looks for environment
while True:

temp = round(sense.get_temperature())
humidity = round(sense.get_humidity())
pressure = round(sense.get_pressure())
message = ' T=%dC, H=%d, P=%d ' %(temp,humidity,pressure)
time.sleep(4)
log = open('weather.txt',"a")
now = str(asctime())
temp = int(temp)
show_number(temp, 200, 0, 60)
time.sleep(5)

if temp >= hot:
pushMessage = "Its hot: " + message
p(pushMessage)
c.perform()
# capture the response from the server
body= buffer.getvalue()
pushMessage = ""

elif <= cold:
pushMessage = "Its cold: " + message
p(pushMessage)
c.perform()
# capture the response from the server
body= buffer.getvalue()
pushMessage = ""

# reset the buffer
buffer.truncate(0)
buffer.seek(0)

# cleanup
c.close()
sense.clear()
GPIO.cleanup()
```