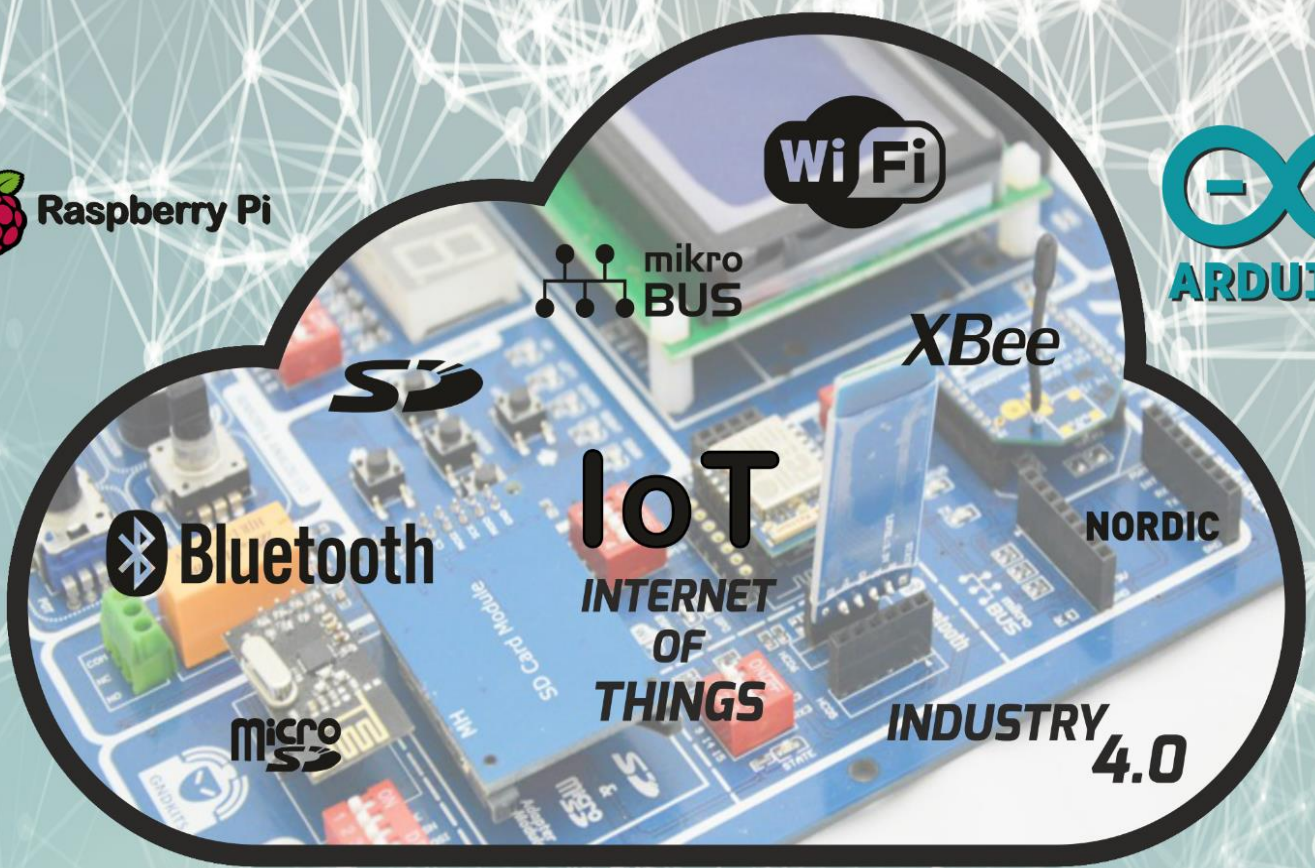
ArduKIT

ARDUINO & RASPBERRY PI DEVELOPMENT KIT

Application Book



GNDKITS

www.ardukit.io

GNDTEKNİK

INTRODUCTION OF EXPERIMENT SET

ArduKIT Experiment Set:

ArduKIT experiment set; It's designed and intended for general use for secondary schools, high schools, colleges, universities and personal hobbies. When making experiment with ArduKIT, you can combine the flexibility of Arduino usage and the functionality of the experiment kit. Because of the shield structure used in applications ready on the ArduKIT, you will make it easier to do the experiment and reduce the material losses when working in the lab.

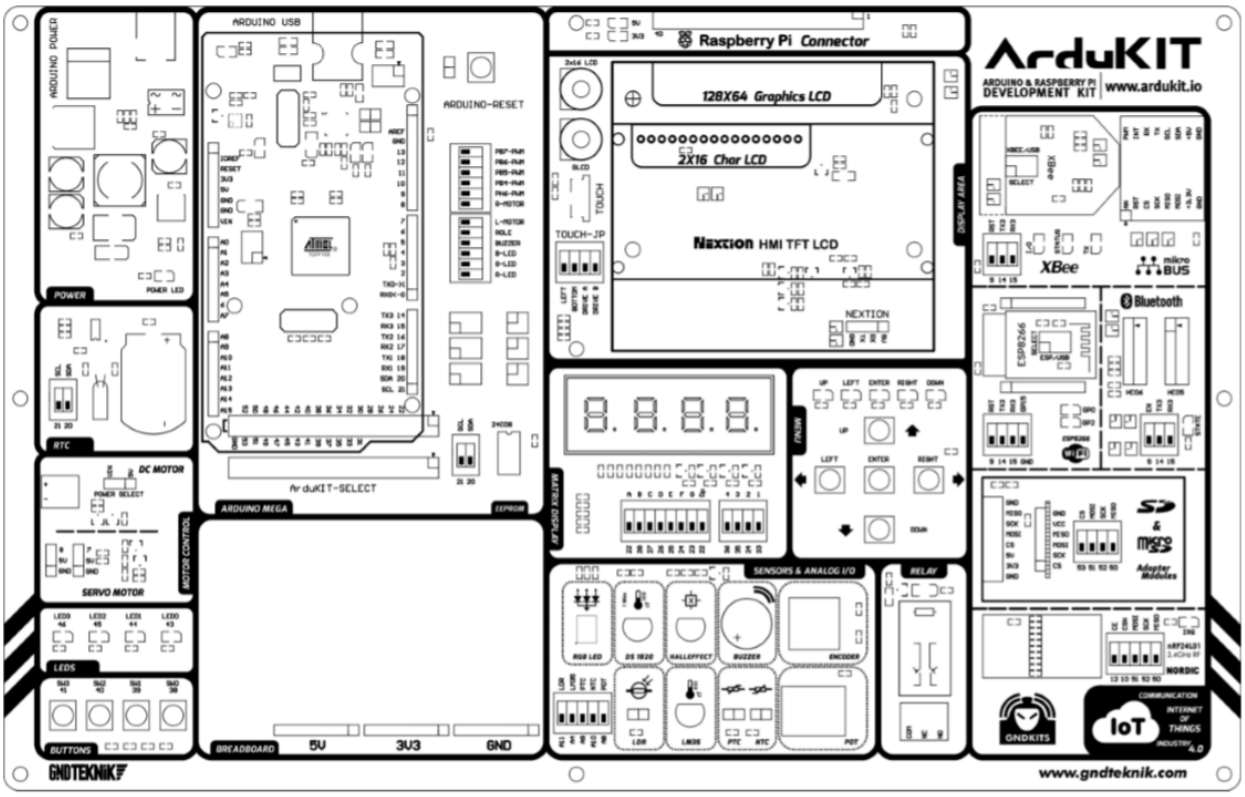
If you use the Raspberry PI 40 pin connector on the ArduKIT training set, you will be able to access most of the circuit elements on the experiment set with Raspberry PI and according to the result when processing the image you will be able to control a physical output (led, motor etc.)

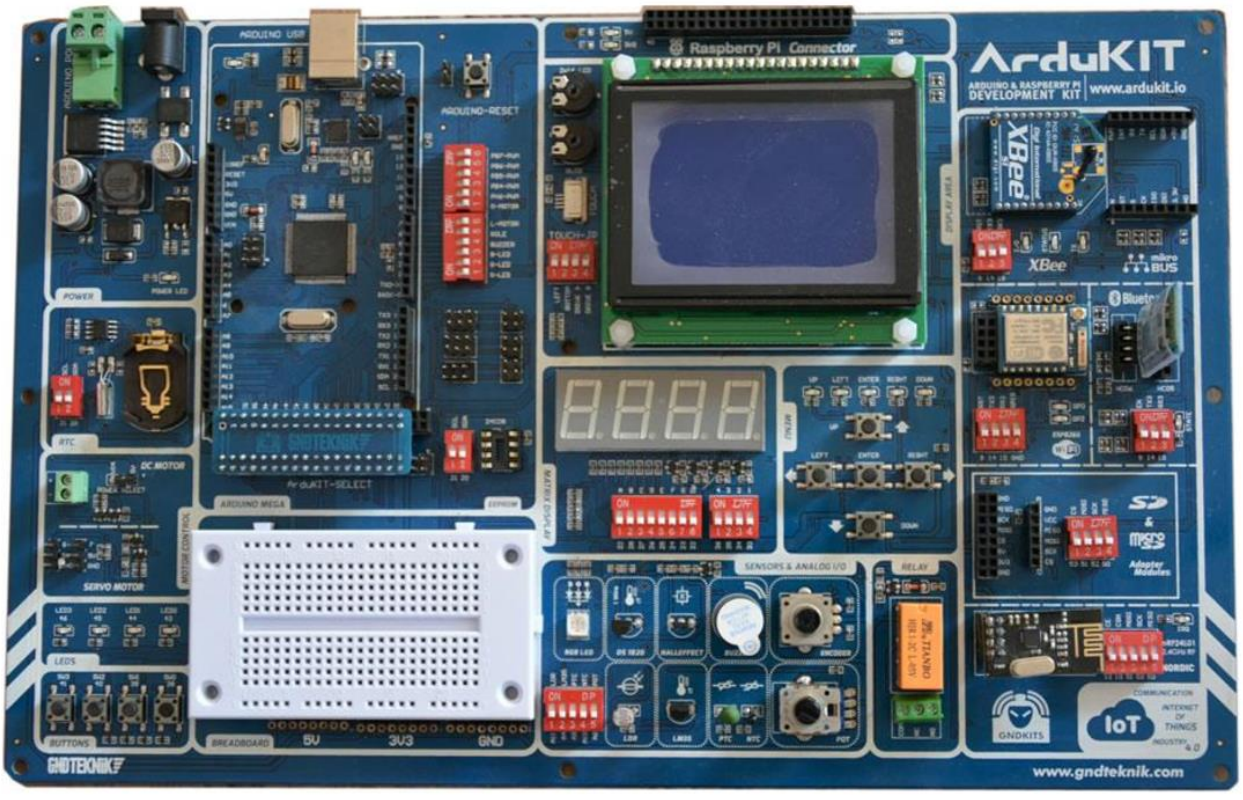
If you wish, you can work with both Arduino and Raspberry PI and combine the flexibility of a 8-bit processor with the power of a 32-bit processor. In all of your experiments, you can also include the PIC series microcontrollers in the your project by using the breadboard on the training set.

In this booklet you will find the general units of the training set and basic applications that can be realized with these units. General information, circuit diagram, procedure, explanation of the program and the list of equipments are prepared for all applications.

Program and library files of all experiments are presented in digital format.

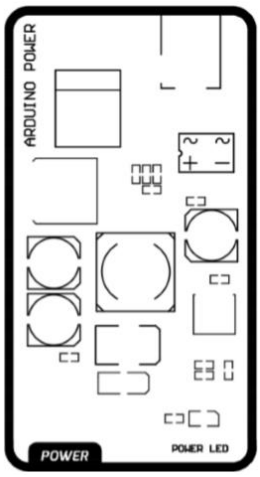
GENERAL OUTLOOK OF EXPERIMENT SET:

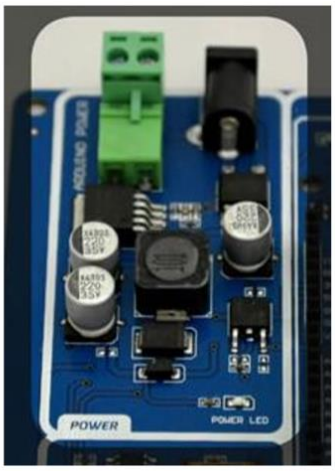




SECTIONOF TRAINING SET

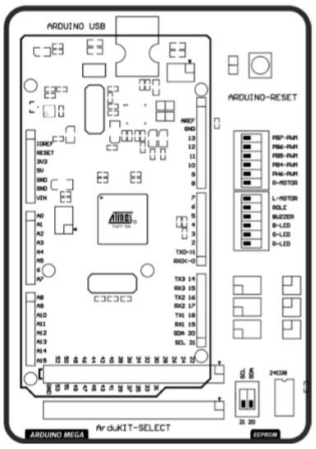
POWER SUPPLY SECTION:





You can apply the energy required for the operation of the experiment set from outside using the power supply module when the energy received from USB port is not sufficient. The external voltage applied from the outside can be from 7 to 30 V DC-AC. In case of AC voltage application, rectifying operation will be done in the circuit.

ARDUINO MEGA 2560 and EXTERNAL EEPROM:

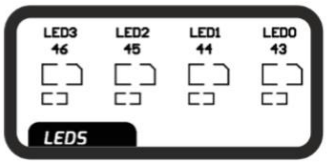


The main control unit of the experiment set is on this block. For to use the input / output peripheral devices on the experiment set with Arduino, connect to the connection bridge pins between ArduKit SELECT pins and Arduino Mega.

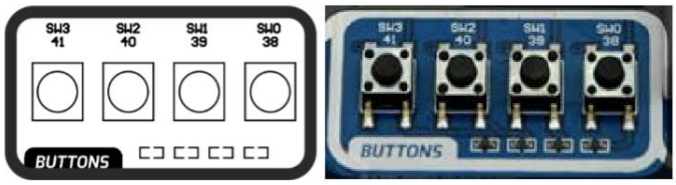
If you wish, you can use the input / output elements with Raspberry Pi. With the EEPROM in this section you can permanently save the data if you want.

The output elements are connected to the corresponding pins and can be disconnected, via the dip switch group in this section.

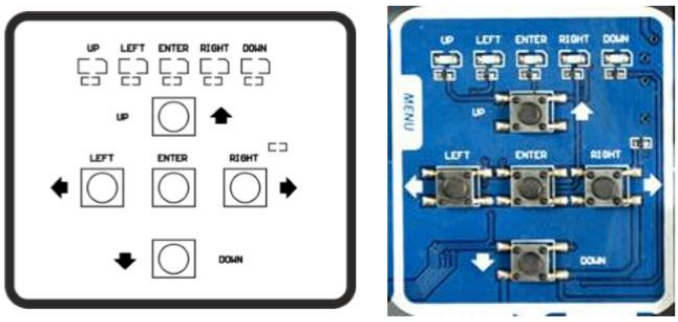
LED INDICATOR SECTION:



INPUT BUTTONS:

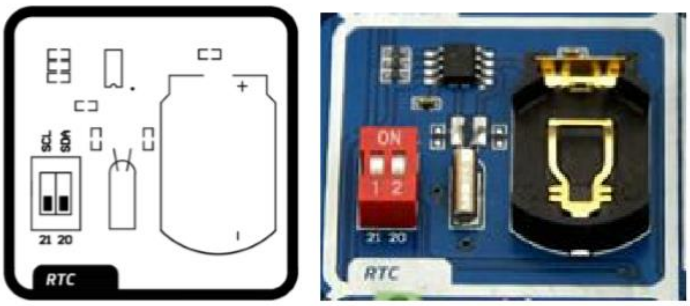


In apps, buttons, as used input elements, directly connected to pins 38-39-40 and 41.



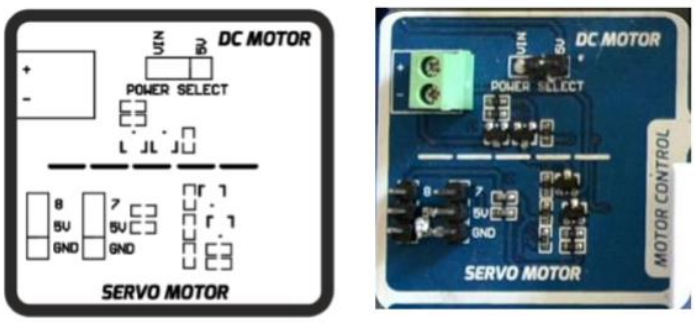
The other button group that you can use for menu applications is connected in parallel to the first button group.

RTC BLOCK:



RTC block contains a real-time clock integration (DS1307) that can communicate with the I2C (SCL-SDA) method that you can use for real-time clock applications.

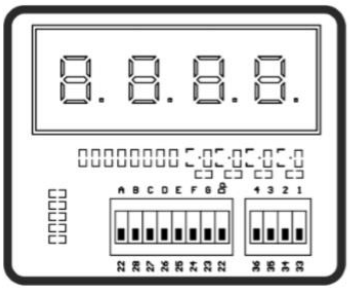
MOTOR DRIVER:

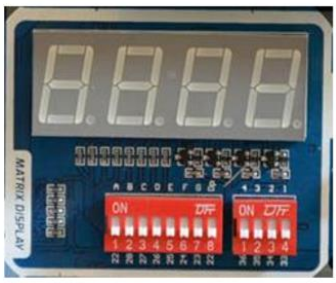


In this section you can make low power DC motor and servo motor applications.

For high-power motors control, you should use different drive circuits.

7 SEGMENT MATRIX DISPLAY:

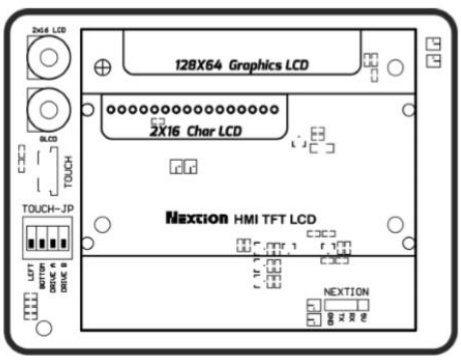


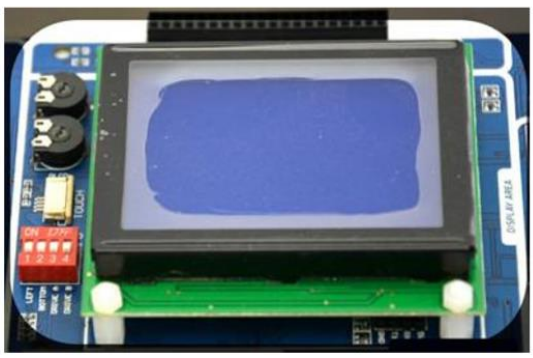


You can make different applications with the 7-segment display section which is one of the most used display elements.

In this section, the data lines providing segment control are connected to the pins as numbered 22,23,24,25,26,27,28,29, and the selection pins which enable energizing for the displays are connected to the pins as numbered 33,34,35 and 36.

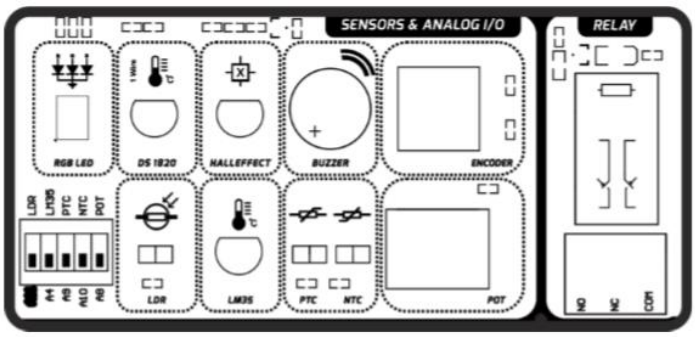
THE SECTION OF DISPLAY:





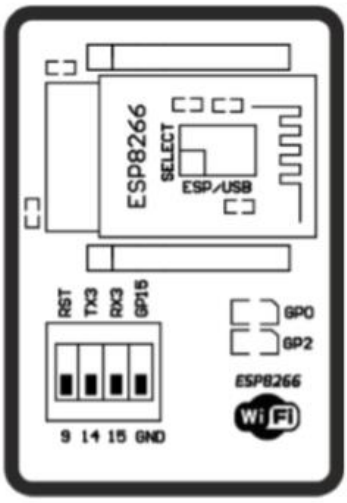
You can use 2x16 character LCD, 128x64 graphic LCD and Touch panel, Nextion HMI TFT touch screen by this part which supports liquid crystal display in different structures. You must set OFF position of the 7-segment display connection while using LCD displays as the 7-segment display overlaps with the data and control pins.

THE SENSORS-BUZZER and POTENTIOMETER:



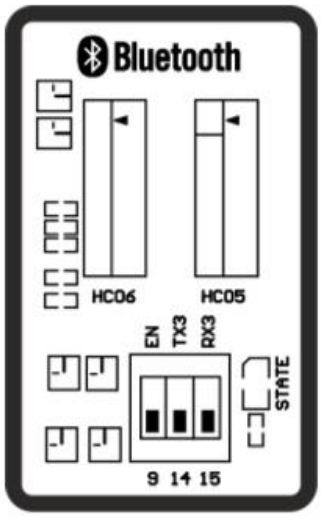
In this section you will find the most commonly used input and output elements. All the elements in this section are connected to separate pins so that you can perform different applications.

THE WI-FI MODULE:



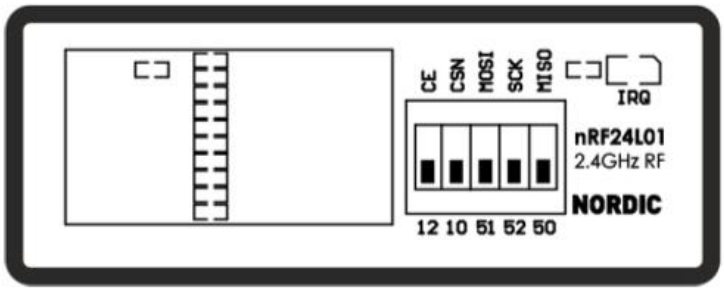
In this section there are ready-made connectors for using the ESP 8266 01 and 07 modules. Using this section you can use the ESP 8266 with AT commands or with the LUA-NODEMCU Firemware update as a standalone microcontroller. You can also use the ESP 8266 microcontroller as if it is an Arduino type by making the necessary updates through the Arduino IDE.

THE BLUETOOTH COMMUNICATION MODULE:



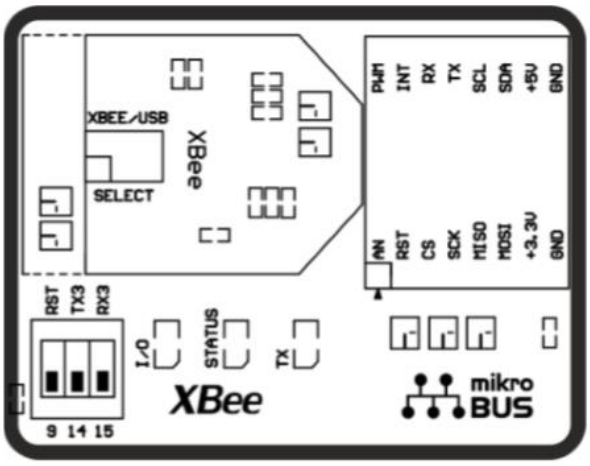
In this section there are the connection pins for the widely used Bluetooth HC05 and HC06 modules. In popular Arduino-Android applications, you can use this section to set up communication link between smart devices and experiment set.

THE RF MODULE:



This section has connection points for nRF24L01 and similar modules developed by Nordic firm. You can link for communication between two experiment sets by RF module.

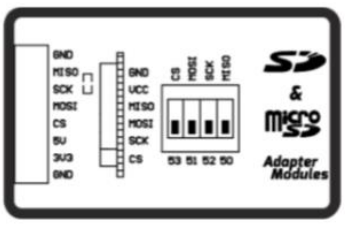
THE XBee MODULE and micro BUS:

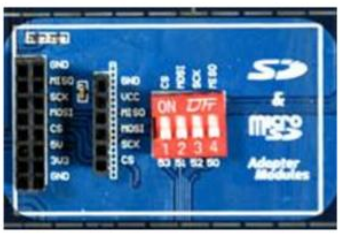


In this section, there is a connection point to use the updated XBee modules. If you wish you can run Xbee modules via Arduino serial communication or directly connect them to your computer. In this case, if you take the Xbee / USB select switch in the ON position, the computer will be disconnect from Arduino MEGA via USB, which it will connect to the Xbee module.

Also in this section there is a connection point for the microClick modules produced by Micro Electronica for easy use.

THE MICRO SD CARD DEPARTMENT:





In this section, there are SD and microSD card connection points that you can use in applications to save measured values or to combine and store different variables.

THE PIN CONNECTIONS:

LEDLER	ARDUINO	RASPBERRY Pi	BUTONS	ARDUINO	RASPBERRY Pi
D1	43	35	B1-RIGHT	38	-
D2	44	38	B2-LEFT	39	-
D3	45	40	B3-FORWARD	40	-
D4	46	15	B4-BACK	41	-
RED	4	-	OK	42	-
GREEN		3		-	
BLUE			2		

7 SEG.DIS	ARDUINO	RASPBERRY Pi	7 SEG.DIS	ARDUINO	RASPBERRY Pi
a	22	07	h	29	12
b	23	29	1	33	35
c	24	31	2	34	38
d	25	32	3	35	40
e	26	33	4	36	15
f		27		36	
g		28		11	

SENSORS	ARDUINO	RASPBERRY Pi	ÇEŞİTLİ	ARDUINO	RASPBERRY Pi
LM35	A4	-	RÖLE	6	-
PTC	A9	-	POTANS.	A8	-
NTC	A10	-	ROT.ENC.	48-49	-
LDR	A11	-	MOTOR SR.	7-8	-
DS18B20	30	-	EEPROM	20-21	-
HALL EFECT	47	-	SD-KART	50-51-52-53	21-19-23-24-26
BUZZER			5		

2x16 LCD	ARDUINO	RASPBERRY Pi	2X16 LCD	ARDUINO	RASPBERRY Pi
D4	26	33	R/W	35	40
D5	27	36	RS	36	15
D6	28	11	BACK LIGHT	31	22
D7	29	12	E	37	16

128X64 GRF.	ARDUINO	RASPBERRY Pi	128X64 GRF.	ARDUINO	RASPBERRY Pi
D0	22	07	BACK LIGHT	31	22
D1	23	29	E	37	16
D2	24	31	R/W	35	40
D3	25	32	RS	36	15
D4	26	33	RST	32	18
D5	27	36	TOUCH SCR.		
D6	28	11	LEFT	A15	-
D7	29	12	BOTTOM	A14	-
CS1	33	35	DRIVEA	A13	-
CS2	34	38	DRIVEB	A12	-

COMMUNICATION MODULS	ARDUINO
NRF24L01	9-10-50-51-52
BLUETOOTH	TX3-RX3-EN:9
X-BEE	TX3-RX3-RST:9
ESP8266	TX3-RX3-RST:9

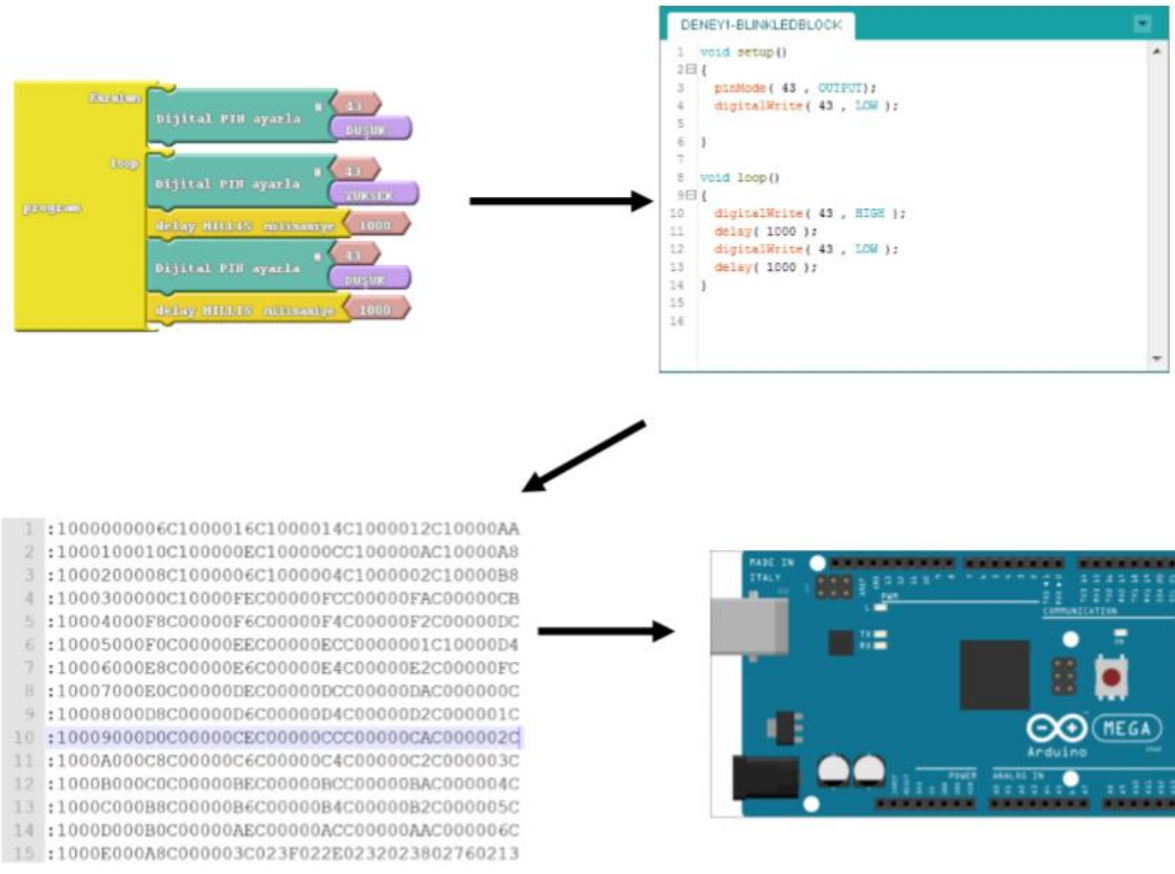
**Since the number of Raspberry Pi I / O is not be sufficient, all elements are not connected in the application set.

THE PROGRAMS USED FOR EXPERIMENTS:

ARDUINO IDE :

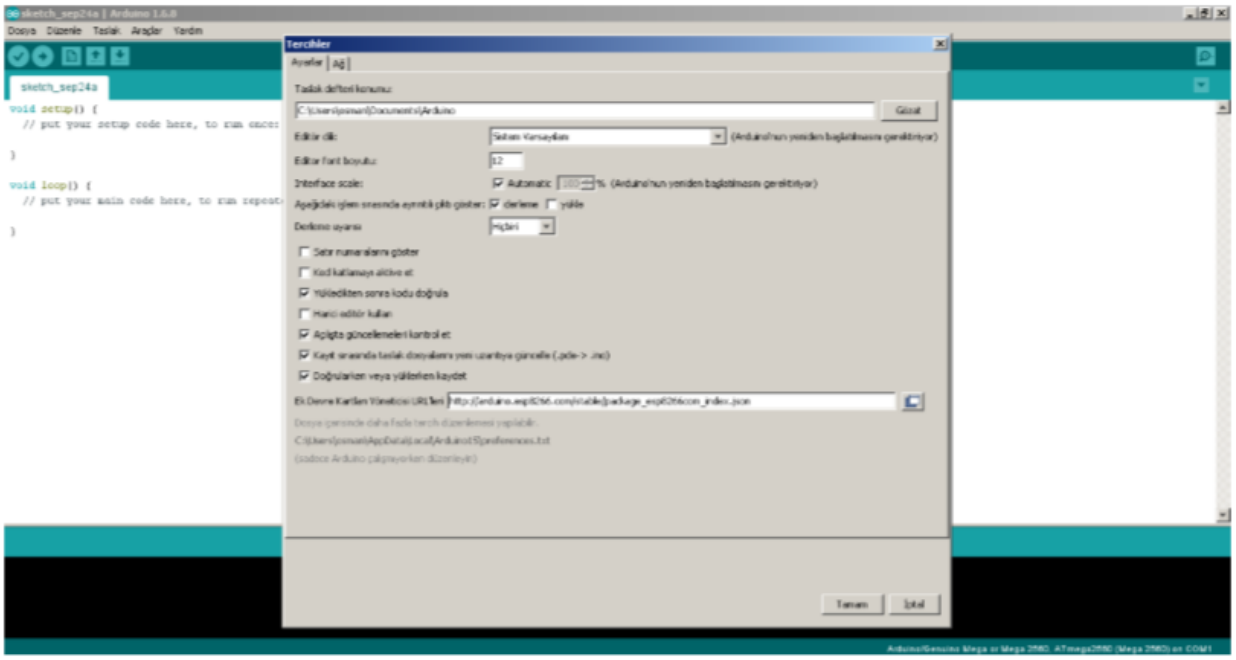
Numerical machines work with machine language consisting of 1 and 0 and can not perceive the words or symbols that we use in everyday life. For this reason, you should plan every step you want from a digital machine and write it in machine language. You need to use editor programs and compilers to write in machine language. You write the operations you want in certain languages (Basic, C, Python i.e…) through editor programs and then translate to machine languages (bin or hex) with compiler programs. You can transfer the file you created in the machine language to the microcontroller with the programmer hardware, and you have completed the application by making the circuit connections.

The Arduino platform provides great convenience here, so you can quickly achieve the result by writing your program and make upload over the same media. If you think that you do not understand the programming rows, you can create your program with symbols using the ArduBlock plugin.



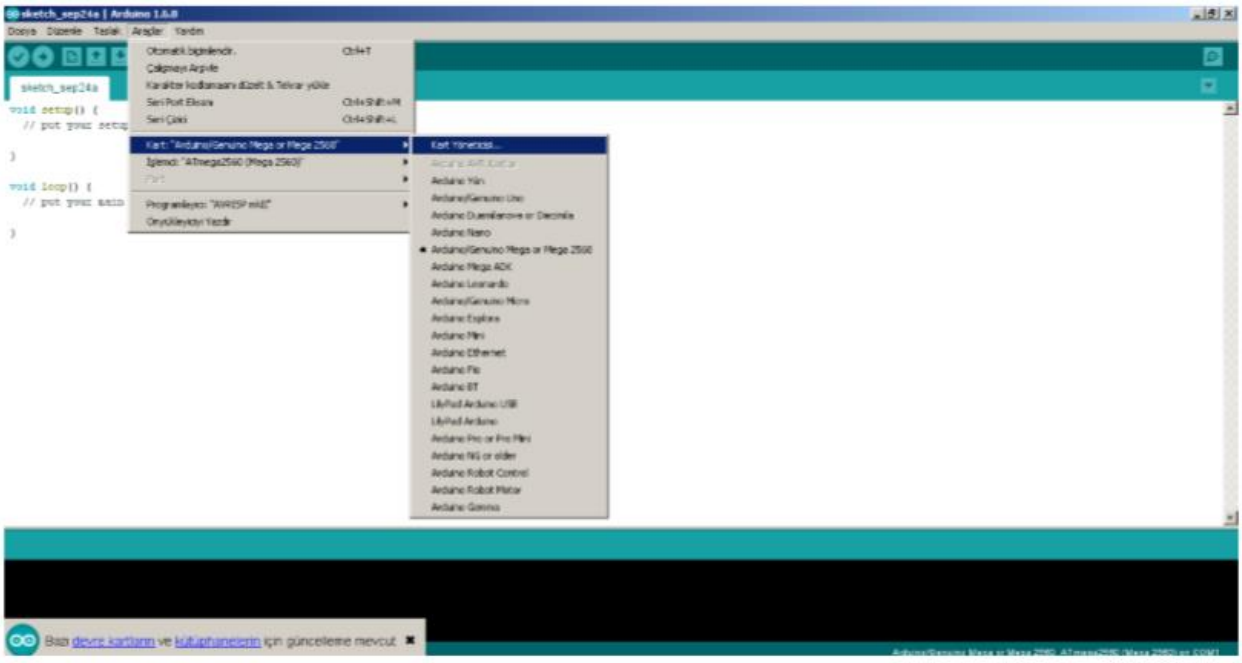
- You can download the latest version of Arduino IDE from www.arduino.cc
- You will need to do search on the internet for the up-to-date download link and installation description of the Ardublock plug-in.

- If you want to program the ESP8266 module directly by the Arduino IDE, click the File-Preferences option on the Arduino IDE. In the pop-up window of the Additional Circuit Cards Manager URLs section you write that link is

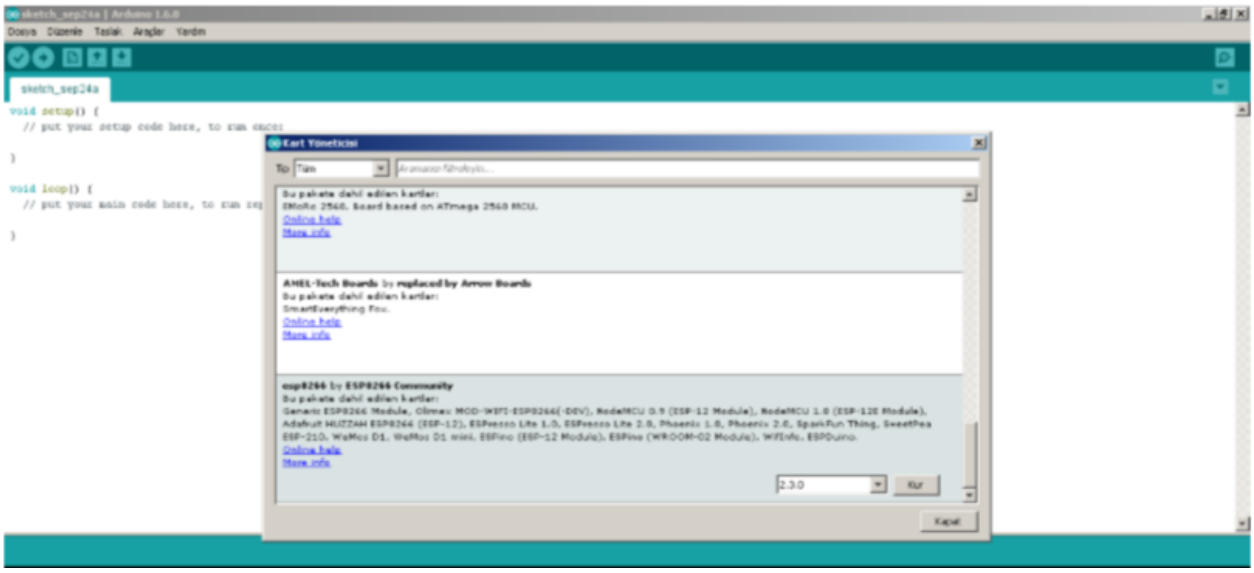


http://arduino.esp8266.com/stable/package_esp8266com_index.json

then click “Update Ok” button. Then on the Tools tab, go to Card and open Card Manager.

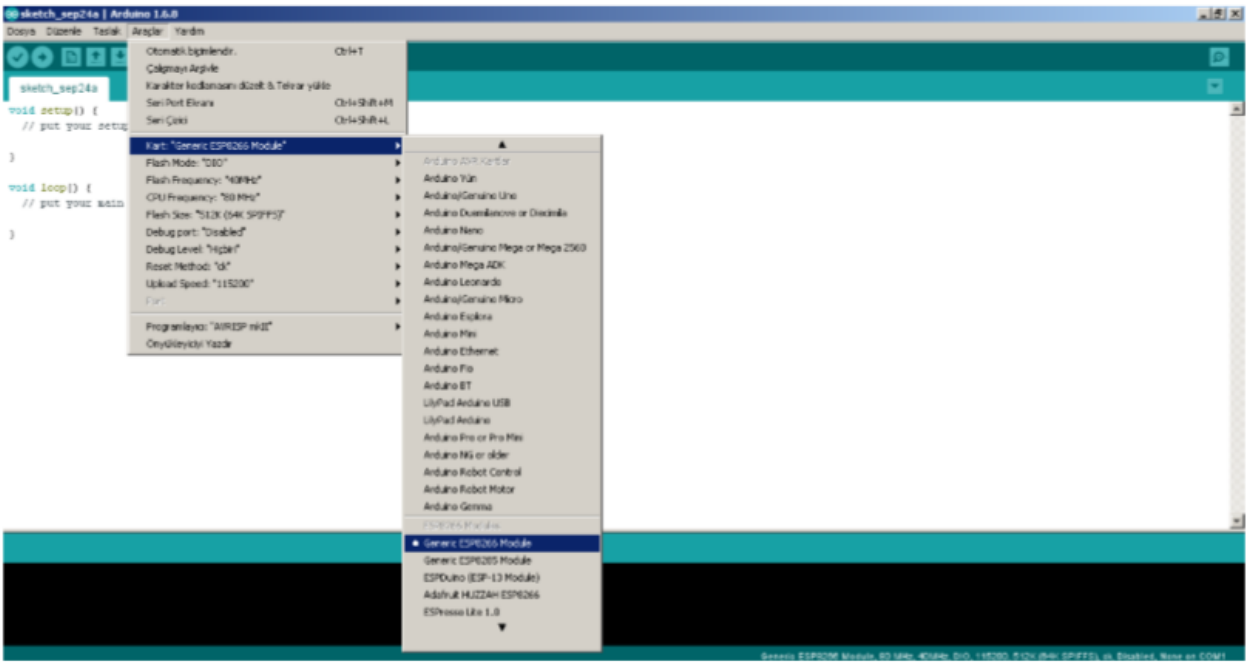


When the Card Manager window opens, it will list updates that can be made with the internet. If you select '' p8266 by ESP8266 Community’’ from the list and click on the Install button, the corresponding package will be installed in the Arduino IDE environment.

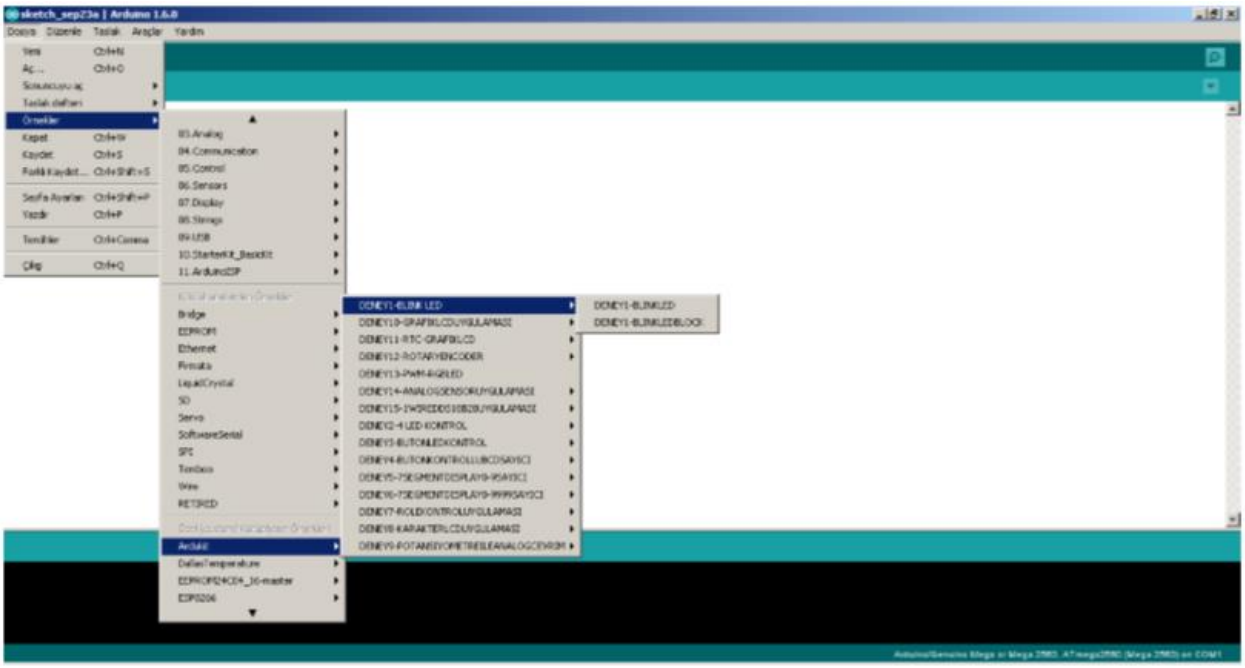


Once the installation is complete, you can program the ESP8266 module with the Arduino IDE directly from the firmware list with the Generic ESP8266 Module option in the list of processors that can be programmed.

**If you have not updated the firmware for the ESP8266 (01-07) module in your hand, you can do this with ArduKIT without needing a different hardware. You will need to do search on the internet for the necessary software and process steps.



*There are also current experiment files in the library files which you can download from www.ardukit.io. If you copy these files to the libraries folder where the Arduino program is installed on your own computer, you can select one ready experiment you want from the experiment list and develop the application like the screen shown below.



WHAT SHOULD BE CAREFULL WHEN MAKE EXPERIMENT?

1- Try to have preliminary information about the operation which you want to apply; your intent must not be just to run the application.

2- There are connections to the circuit diagrams and the simulation program in the preliminary information part of the basic level test pattern for you can visualize the connection. Do not try to set up the circuit unless specified in the circuit diagram, the necessary connections are made on the experiment set.

3- The program can be written with the drawing made by the ArduBlock plugin for basic operations. However, for complex experiments, the ArduBlock symbols are inadequate and that which either unnecessarily crowded with the process. For this reason, you can find the program files made with the ArduBlock plugin for basic experiments in the corresponding experiment folder, even if it is not for all experiments.

4- The experiment sheets were prepared for a large community. For this reason, the preliminary information part may seem unnecessary or insufficient for you, in this case, be careful to research from different sources.

5- The lines of code found in the experiment sheets may not be the complete program, please examine the test file from the relevant test folder.

6- At the beginning of the program files there are basic explanations about the experiment, the source code addresses where the program codes are inspected and the internet addresses where you can find the latest version of the relevant library files. You can access the original codes from the relevant addresses.

7- Since the experiments were prepared for training purposes, they were created from basic to difficult level. If you have an application that it can not run, or you do not understand, do not hesitate to check out previous applications and do not think of it as a waste of time.

8- Do research, for to be able to solve the exercises given at the end of the experiment with different methods.

9- You can visit www.ardukit.io for current and/or advanced experiments not included in the book. If you wish, you can earn small gifts by sharing your original work with us.

10- Please note that for your experiments with Raspberry Pi, the operating system and additional packages must be installed on SD card. You can make applications such as led control, use of LCD screen, serial communication via USB connection with the examples in the book.

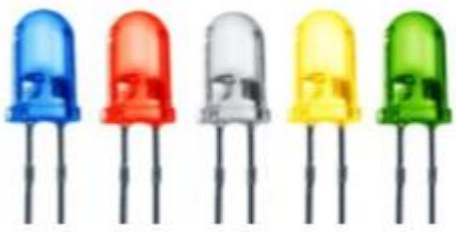
11- The hardware UART connection is not working properly, especially as a result of the bluetooth module conflict in Raspberry Pi 3 model. This problem can be solved with software updates for Raspberry Pi in the future. You can use the Bluetooth module for wireless communication applications with ArduKit.

12- Remember to energize the micro USB input with 5V (1A - 2A) adapter according to the model you are working with Raspberry Pi.

TEST NAME: THE CONTROL OF THE LED **TEST NO:1**

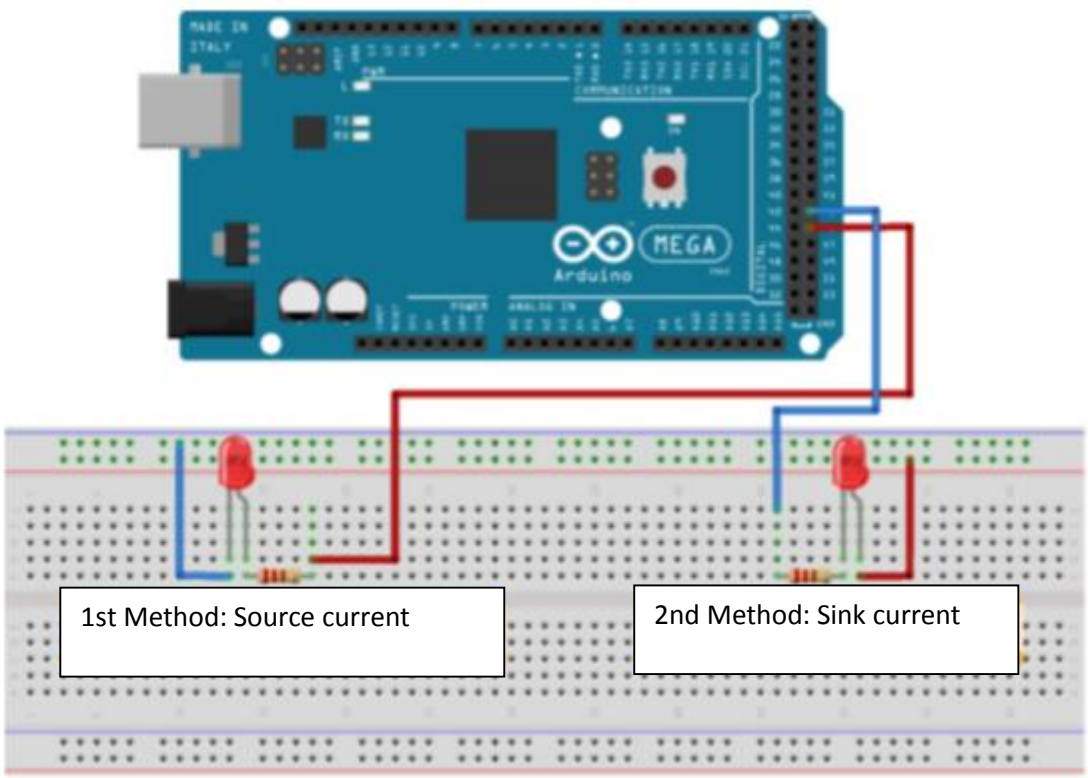
OBJECTIVE: Having knowledge about the Arduino IDE environment that we used during the experiments and apply control of the led by connecting to the ArduKIT.

PRELIMINARY INFORMATION:



The LED we use as the representative load in control applications is a semiconductor element and can be operated in two ways with a series connected resistor. In the first method, the LED is connected to the cathode (-) ground level, and the control signal is connected to the anode (+) terminal through the resistor, and when the signal is 1, the LED is on, when the signal is 0, the LED is off.

In the second method, the LED is connected to the AOT1 level (5V or 3.3V) and the control signal is connected to the Cathode terminal through the resistor, and the signal is 0 when the LED is on and when the signal is 1 the led is off.



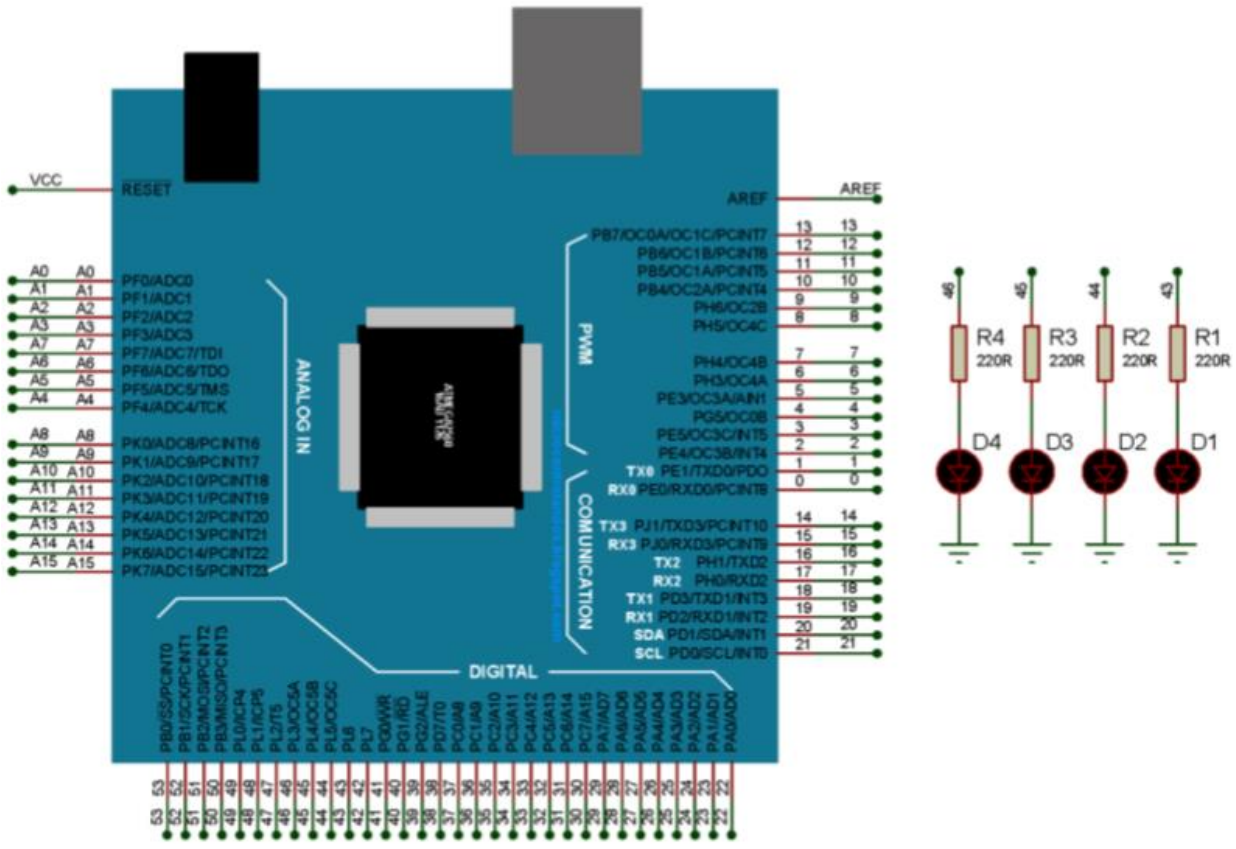
The output connections on the test set were applied by the first method. In other words, you must set the output terminal (Led-Relay-Motor-7 Segment Display etc.) you want to run to Logic 1 (high). The pins to be used as output in the Arduino IDE environment should be specified in the setup function.

If you have already worked with C-based programming languages for the software part, you can complete the applications without difficulty. Since the commands used in the ArduinoIDE environment are prepared with C ++, you need to be careful of big and small letters when using commands. If you think it's hard for you to write the code, perform the basic functions with ArduBlock and load directly into ArduKIT. ArduBlock will do the conversion process automatically.

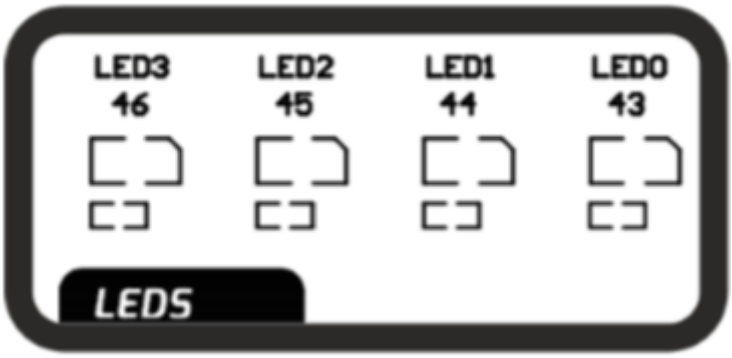
PROCESSING STEPS:

1- Install the program on Arduino, by writing it in ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.
4- Make necessary changes on the program so that the other LEDs in the experiment set will blink at 2 second intervals.

CIRCUIT DIAGRAM:



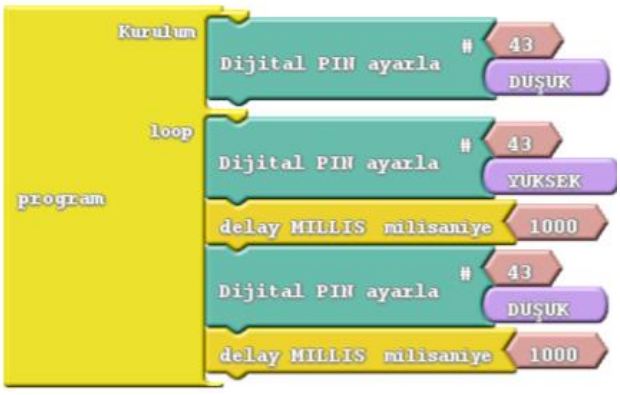
,



PROGRAM CODE:

```
// Pre-settings are made for the pins and processor used with the setup function.

void setup() {

pinMode(43, OUTPUT);// Pin 43 is set as output

}

// Loop function is the section where that do the operations we want to run.

void loop() {

digitalWrite(43, HIGH); // LED is on (HIGH allows the voltage level to be logic 1 or 5 V)

delay(1000);                // Wait 1 second (We write in milliseconds)

digitalWrite(43, LOW);  // LED is off (Allows LOW voltage level to be logic 0 or 0 V)

delay(1000);                // Wait 1 second (We write in milliseconds)

}
```

ArduBlock Drawing:



It will be easier to understand because there is support of even Turkish language on ArduBlock. In the program we created with the symbols, after setting the pin in the setup section, 43th pin in the loop section in that case 1th led will flash at 1 second intervals.

TEST NAME: The Apply Of Knight Rider With Four Leds **TEST NO:2**

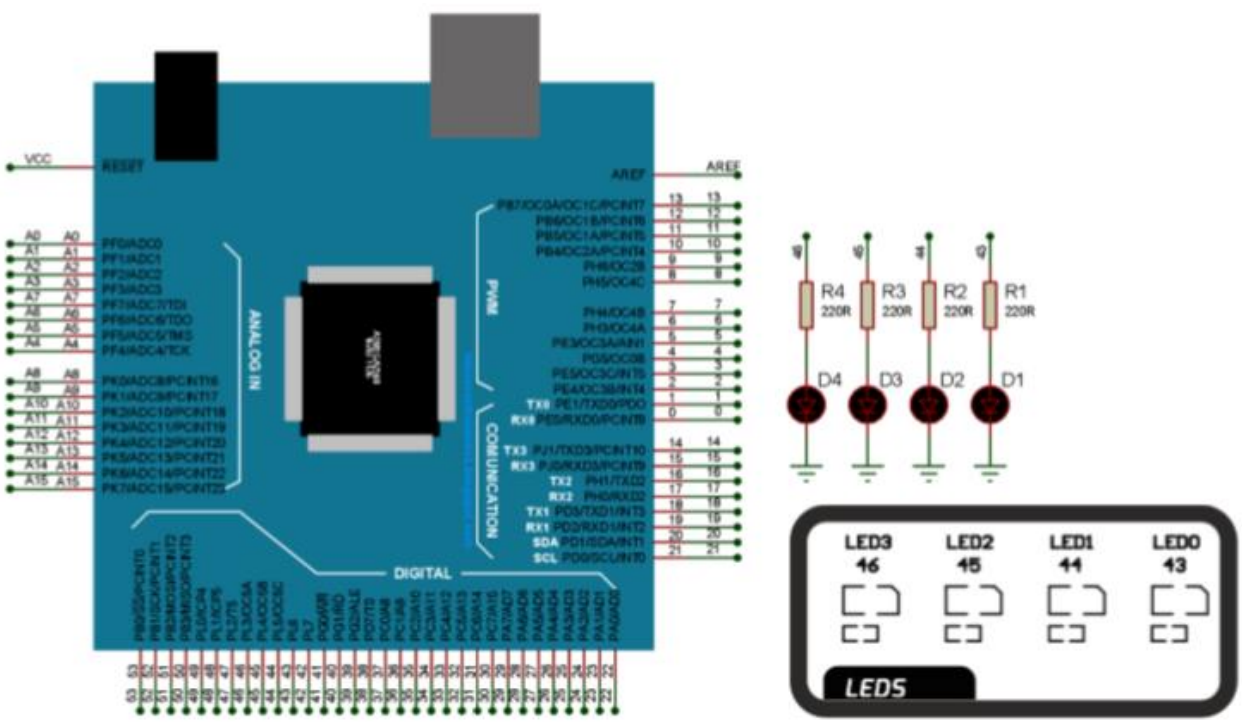
OBJECTIVE: The take control of multiple outputs with loop.

PRELIMINARY INFORMATION: When you want to check out more than one output, we perform this operation by loading byte values into ports in classical microcontrollers but with Arduino simplicity, we have to work with pin logic, which is the biggest disadvanta age, and we do not know connecting that pins with which ports of the processor at back ground. For example, Pin 43, which we used in the first application, is connected to 6 bits (L7 ... L0) at L port for Atmega2560 processor. For this reason, we perform the control operation on the Arduino platform by making the desired pins HIGH (1) or LOW (0) in the cycle.

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.
4- Make the necessary changes on the program so that the LEDs blink in 2 second intervals as knight rider. The application works in the form of walking light, and when you reverse the process with the last led burning; it will take the arrangement of the knight rider.

CIRCUIT DIAGRAM:



PROGRAM CODE:

```
// Pre-settings are made for the pins and processor used with the setup function.

void setup() {

// Pins 43-44-45-46 are set as output

pinMode(43, OUTPUT);
pinMode(44, OUTPUT);
pinMode(45, OUTPUT);
pinMode(46, OUTPUT);
}

// Loop function is the section where that do the operations we want to run.

void loop() {

digitalWrite(46, LOW);          // 4. LED is off (Allows LOW voltage level to be logic 0 ie 0 V)

digitalWrite(43, HIGH);         // 1. LED is on (HIGH allows the voltage level to be Logic 1 or 5 V)

delay(1000);                    // Wait 1 second (We write in milliseconds)

digitalWrite(43, LOW);          // 1. LED is off. (Allows LOW voltage level to be logic 0 ie 0 V)

digitalWrite(44, HIGH);         // 2. LED is on. (HIGH allows the voltage level to be Logic 1 or 5 V)

delay(1000);                    // Wait 1 second (We write in milliseconds)

digitalWrite(44, LOW);          // 2. LED is off. (Allows LOW voltage level to be logic 0 ie 0 V)

digitalWrite(45, HIGH);         // 3. LED is on. (HIGH allows the voltage level to be Logic 1 or 5 V)

delay(1000);                    // Wait 1 second (We write in milliseconds)

digitalWrite(45, LOW);          // 3. LED is off. (Allows LOW voltage level to be logic 0 ie 0 V)

digitalWrite(46, HIGH);         // 4. LED is on. (HIGH allows the voltage level to be Logic 1 or 5 V)

delay(1000);                    // Wait 1 second ((We write in milliseconds)
}
```

ArduBlock Drawing:

When drawing with ArduBlock, you do not have to installation setting for basic input-output operations. When you press the Load button on the Arduino, the program will automatically create setup and loop functions.

TEST NAME: The Control of The Led With Button **TEST NO:3**

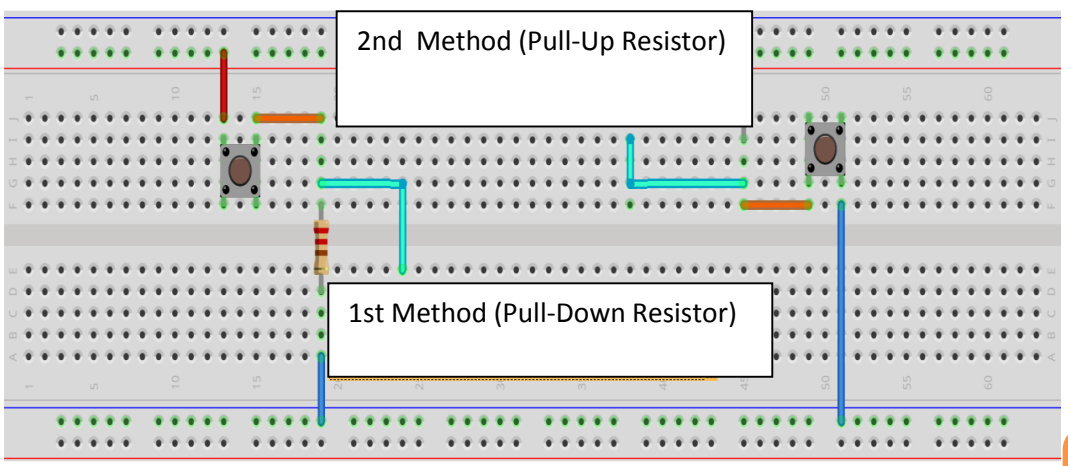
OBJECTIVE: Apply Input-Output with Arduino, learning decision expression

PRELIMINARY INFORMATION:

As the output element we use the LEDs symbolically as well as the input elements we use the buttons. The input operation performed by the button may be dependent on variables such as the key (the button is sensor in real life). We can connect the input element in two ways as it is in the output operation. In the first method, one terminal of the button is connected to the HIGH (logic 1) level and the other terminal is connected to the desired pin. In this case, the relevant pin is connected to a resistor called PULL-DOWN resistor, which pulls the relevant pin to logic-0. So, when the buton is pressed, the logic level is HIGH, and when it is released, it becomes LOW (0).

In the second method, one terminal of the button is connected to the LOW level and the other terminal is connected to the desired pin. . In this case, the relevant pin is connected to a resistor called PULL-UP resistor, which pulls the relevant pin to logic-1. So, when the buton is pressed, the logic level is LOW (0). When it is released, it becomes HIGH.

<u>If the pull-up and pull-down resistors are not connected and the button is not pressed, the relevant pin will be idle and the application will run unsteady .</u>

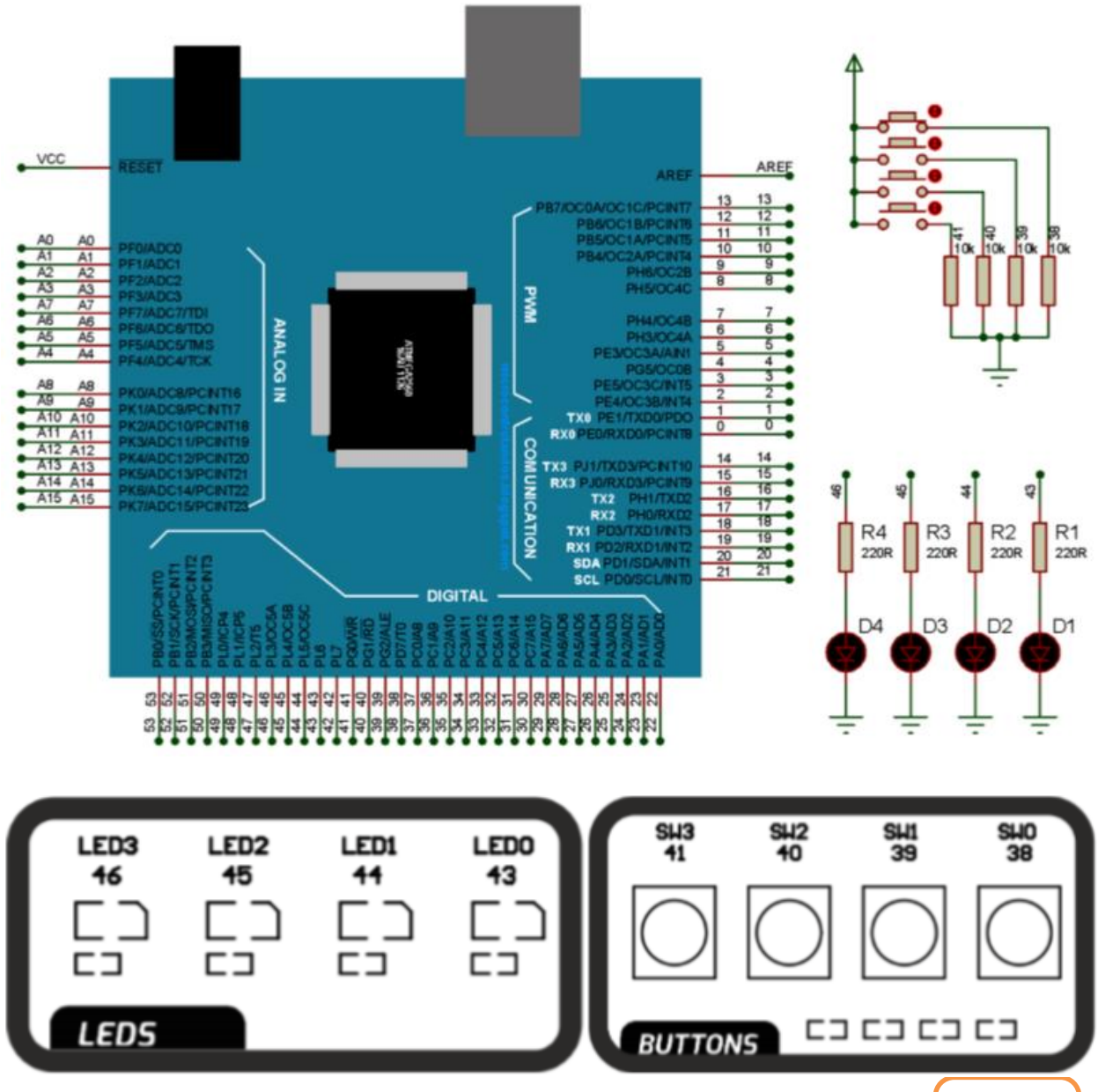


The input connections on the test set were applied by the first method. That is, when the button is pressed, the relevant pin is HIGH level, and when it is released, it is LOW level. In practice, pull-up and pull-down resistors are selected from 1K to 10K.

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.
4- By making the necessary changes on the program, you can make all of the leds light when first button is pressed, all of the leds off when second button is pressed.

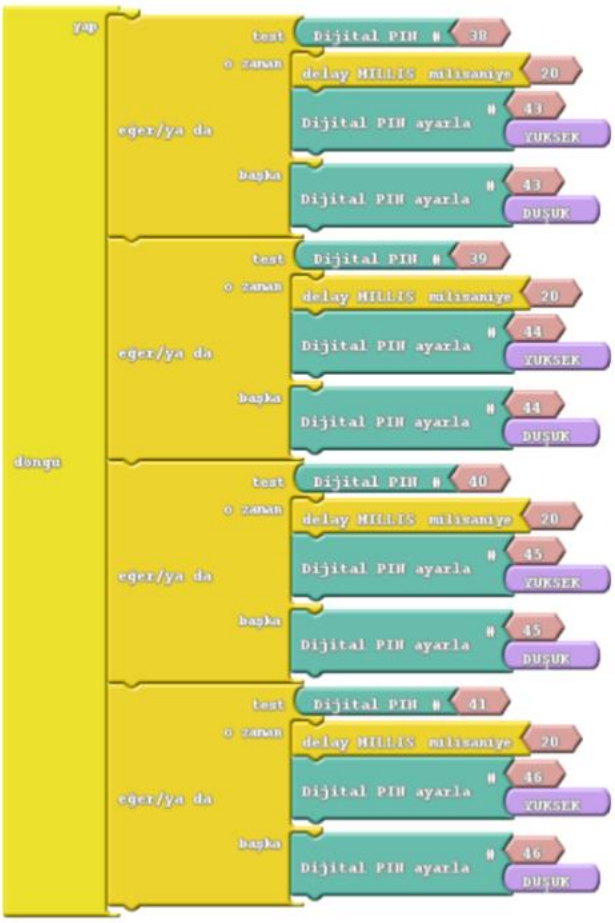
CIRCUIT DIAGRAM:



PROGRAM CODE:

```
// The used constants in the program are defined in this section.

    const int leds[]={43,44,45,46};
    const int buttons[]={38,39,40,41};

// The used variables in the program are defined in this section.

    int i;

// Pre-settings are made by setup function for the used pins and processor

    void setup() {

// Pins 43-44-45-46 are set as output, and pins 38-39-40-41 are set as input

        pinMode(leds[0], OUTPUT);
        pinMode(leds[1], OUTPUT);
        pinMode(leds[2], OUTPUT);
        pinMode(leds[3], OUTPUT);
        pinMode(buttons[0], INPUT);
        pinMode(buttons [1], INPUT);
        pinMode(buttons [2], INPUT);
        pinMode(buttons [3], INPUT);
    }

// Loop function is the section where that do the operations we want to run.

void loop() {

// Reads the value of the pressed button and makes the relevant LED pin HIGH or LOW.

for(i = 0; i <4; i ++)                    //Executes a loop from 0 to 3, depending the variable of i

{if(digitalRead (buttons[i]==HIGH)){      // If the relevant button's pin is 1, for i value
delay(20);
digitalWrite(leds[i],HIGH);}              // it waits 20 ms and makes the relevant  LED's pin 1.
else                                      // If the relevant button's pin is not 1, for i value
digitalWrite(leds[i],LOW);                 // makes the relevant LED's pin 0

}

}
```

ArduBlock Drawing:



ArduBlock drawing is more understandable than writing program's with Arduino IDE. Although it is longer than the program which is writing with Arduino IDE.

A wait period of 20 ms was added to wait for the electric noise to be lost which was generated when the button press.

When there is no waiting period, Although not critical for this application, the program can work as if it were printed more than once on a button.

TEST NAME: BUTTON CONTROLLED BCD COUNTER **TEST NO:4**

OBJECTIVE: To practice BCD counter by learning the input-output application and decision statement logic.

PRELIMINARY INFORMATION:

Numerical values on numeric machines do not progress on a number scale, as we have seen in mathematics lessons, from minus infinite to plus infinite. It takes a value up to the upper bound of the variables we have defined. For example, the variable one byte (8 bits) can have take 256 different values, with a minimum of 0 (00000000) and a maximum of 255 (11111111). In the same way, if a 16-bit variable is unsigned, it can take a maximum of 65535. If the highest weighted bit is used as the sign bit the range of values that it can take between -32767 ... + 32767.

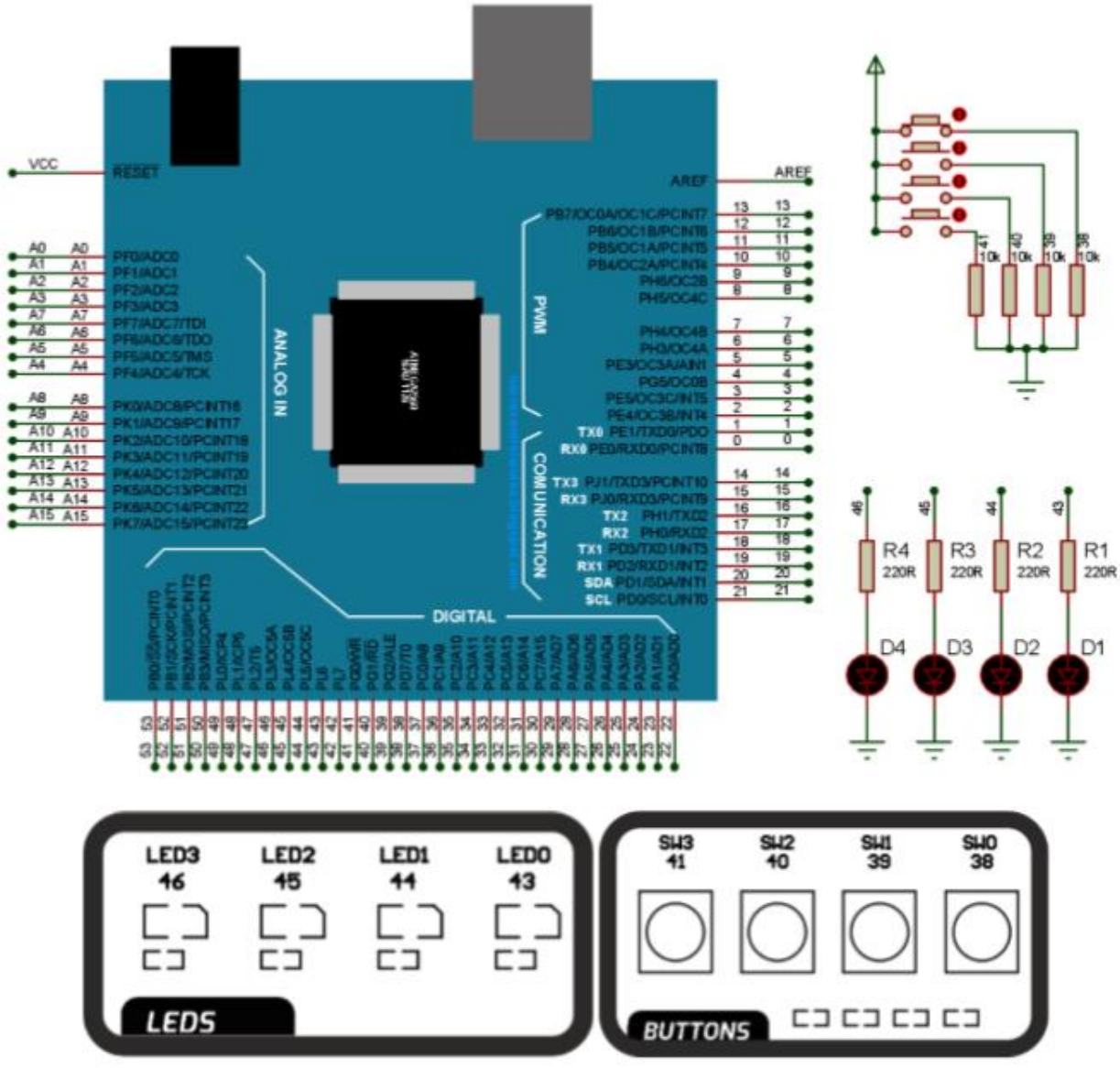
BCD (Binary Coded Decimal) values are the numbers that are used in daily life and are coded in the binary number system and each digit is represented by 4 bits. 4 LEDs on the ArduKIT will show these 4 bits. In the table below, the first 10 values are equal to the BCD equivalent of the numbers 0-9, the hexadecimal (hexadecimal) number system has been developed to shorten binary numbers. If you remember, we loaded the file with the HEX extension consisted of the characters from 0 to F on the ArduKIT during the experiments.

DECIMAL	BINARY	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.
4- By making the necessary changes on the program, runs the counter for it counts back and forth of between 0-15 (0000-1111).

CIRCUIT DIAGRAM:



PROGRAM CODE:

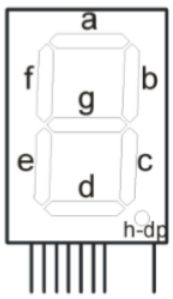
```
// The used constants in the program are defined in this section.

// Loop function is the section where that do the operations we want to run.

void loop() {

// We will increase or decrease the value variable according to the pressed button.

// We will show the BCD count value between 0 and 9 with LEDs.

if(digitalRead(plus)==HIGH)       // If the plus button is pressed

{
delay(20);                        //  20 ms  waits for electrical noise to be lost.

value= value+1;                   //  Increases the value variable by one

if(value >=10) value =0;          //  If the value is equal to 10,that is after 9, the result is 0

while(digitalRead(plus)==HIGH); //  When the button is pressed, it waits release for the value to be
changed once.
}
if(digitalRead(minus)==HIGH)    //  If the minus button is pressed
{
delay(20);                        //  20 ms ms  waits for electrical noise to be lost
value = value -1;                 //  Decreases the value variable by one
if(value ==-1) value =9;          //  If the value is equal to -1, that is before 0, the result is 9
while(digitalRead(minusi)==HIGH);    //  When the button is pressed, it waits release
                                  // for the value to be changed once.
}
switch(value){                    // Determines the LEDs to be light or not light  according to
                                  //the content of the value variable
case 0 : // value==0 ise leds=0000
digitalWrite(leds[0],LOW);
digitalWrite(leds[1],LOW);
digitalWrite(leds[2],LOW);
digitalWrite(leds[3],LOW);
break;
-
-
-
case 9 : //value==9 leds=1001
digitalWrite(leds[0],HIGH);
digitalWrite(leds[1],LOW);
digitalWrite(leds[2],LOW);
digitalWrite(leds[3],HIGH);
break;
}
}
```

TEST NAME: 7 SEGMENT 0-9 COUNTER **TEST NO:5**

OBJECTIVE: To learn the working logic of commonly used 7-Segment display and to implement a sample application.

PRELIMINARY INFORMATION:

The seven segment display indications are two types, common cathode and common anode. There are four displays in common cathode structure in the test set. 0 or 1 is sent to the segments according to the characters to be show on the display. When segments are named, the upper character is named clockwise as a segment. Segments are shown in lowercase so that they are not confused with port numbers in microelectronics with input values in digital electronics.



For example, to show digit 7 on the common cathode display, logic-1 applied to the a, b, and c segments and logic-0 applied to the other segments. In the test set, the a segment was set to 29th pin, the h segment to 22nd pin, and the others to settle (29-28-27-26-25-24-23-22) respectively.

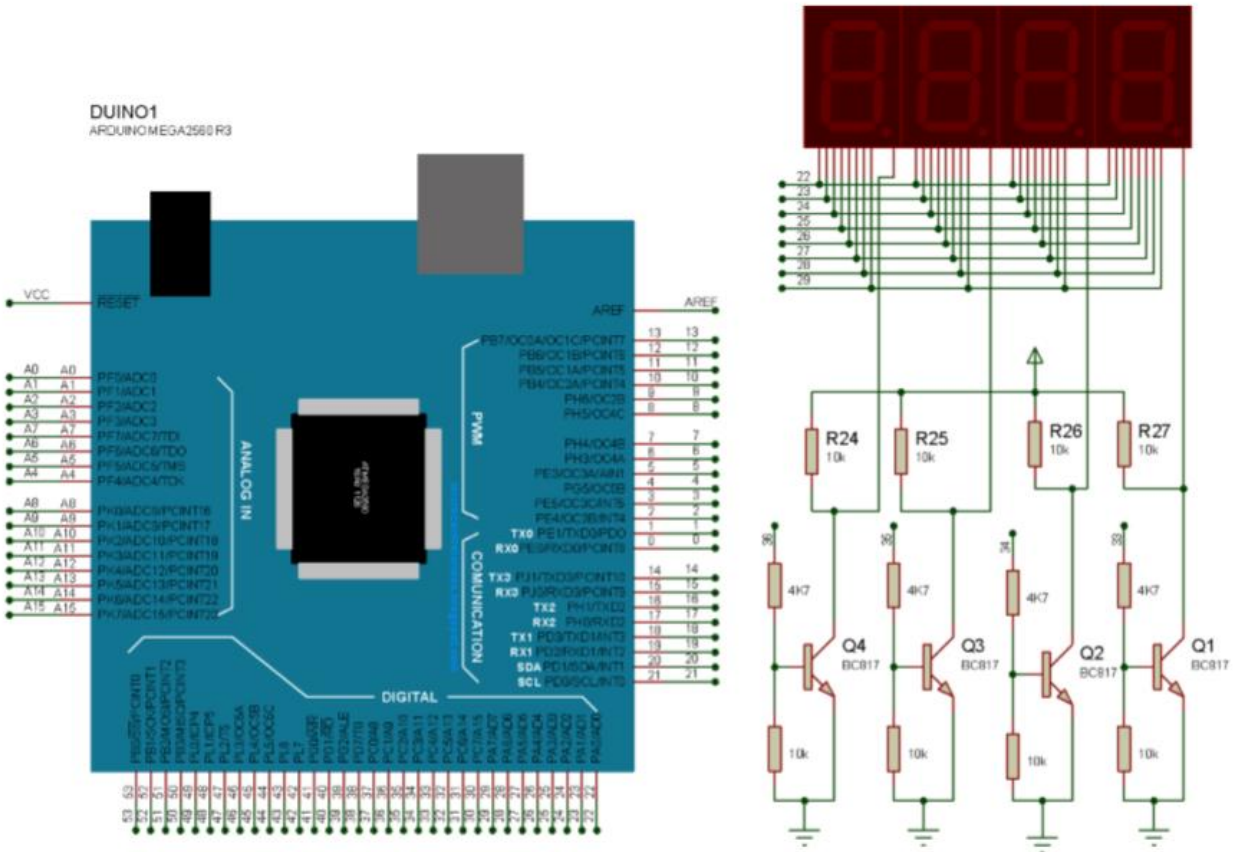
In practice, when more than one display is used, desired to be run the display's a common cathode terminal or a common anode terminal is driven with a transistor by using a common data line.

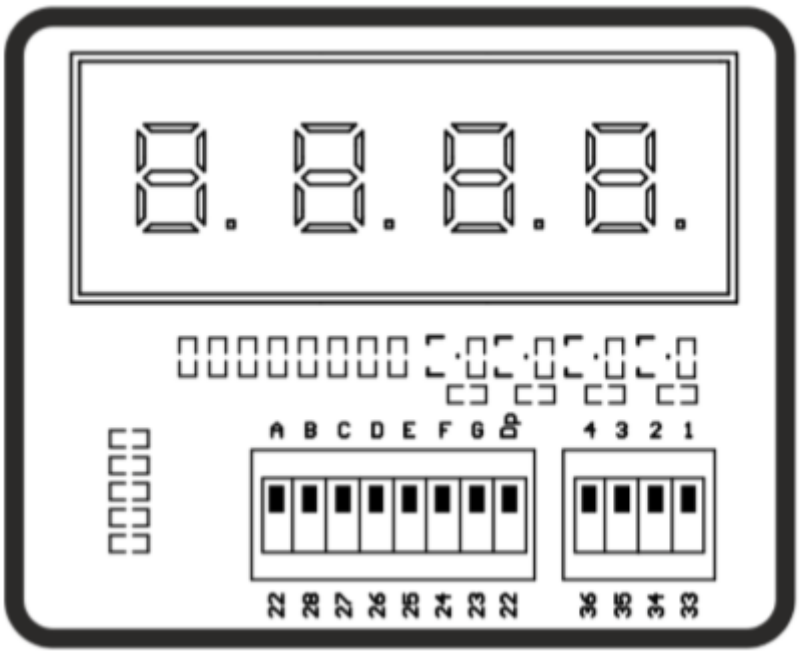
So that the segment terminals do not occupy the individual pins. It is also a very prevalent application to drive segment terminals with D flip flop integrations like 74HC595. The transistors that activate the common cathode terminals in the test set are connected to pins 33-34-35-36.

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.
4- By making the necessary changes on the program, run the counter for it count between 0-99.

CIRCUIT DIAGRAM:





PROGRAMME CODE:

```
int aPin = 22;                  //                A
int bPin = 23;                  //              _____
int cPin = 24;                  //         |          |
int dPin = 25;                  //    F    |          |    B
int ePin = 26;                  //         |          |
int fPin = 27;                  //         |_____     |
int gPin = 28;                  //         |  G       |
int hpin = 29;                  //         |          |
int GND1 = 33;                  //         |          |
int GND2 = 34;                  //    E    |          |    C
int GND3 = 35;                  //         |_____     |  o DP(H)
int GND4 = 36;                  //               D
int dig1 = 0;
int arti = 38;
int eksi = 39;
void setup()
{
pinMode(aPin, OUTPUT);
.
.
.
}
}
void clearLEDs()
{
digitalWrite( 2, LOW);          // A
digitalWrite( 3, LOW);          // B
digitalWrite( 4, LOW);          // C
digitalWrite( 5, LOW);          // D
digitalWrite( 6, LOW);          // E
digitalWrite( 7, LOW);          // F
digitalWrite( 8, LOW);          // G
}
.
.
.
```

TEST NAME: 7 SEGMENT 0-9999 COUNTER **TEST NO:6**

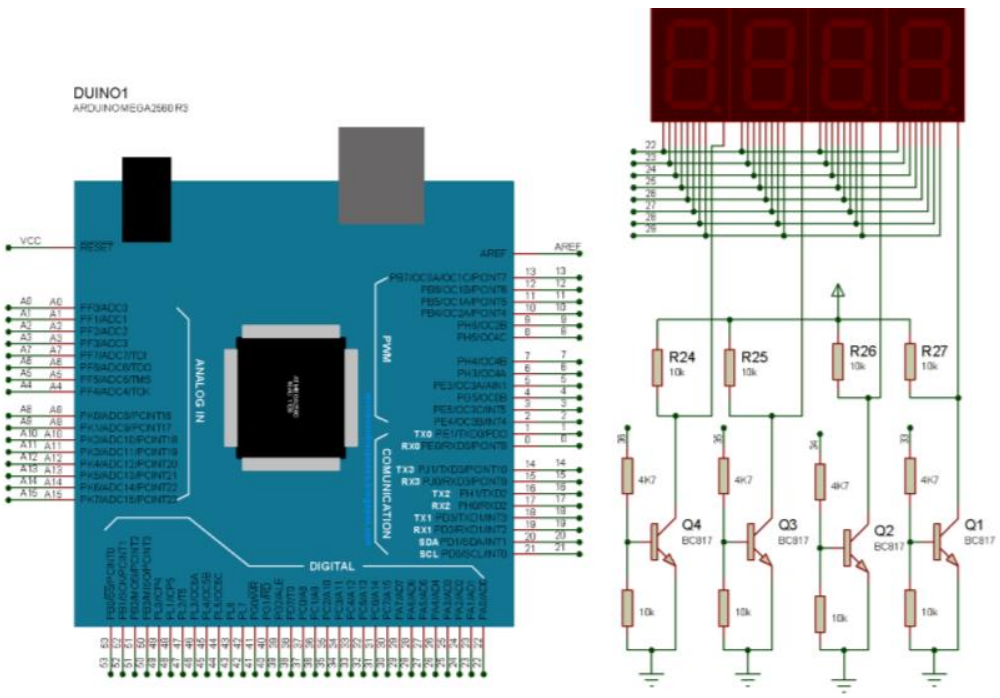
OBJECTIVE: To learn how to run more than one 7 segment indicators in matrix structure and implement them.

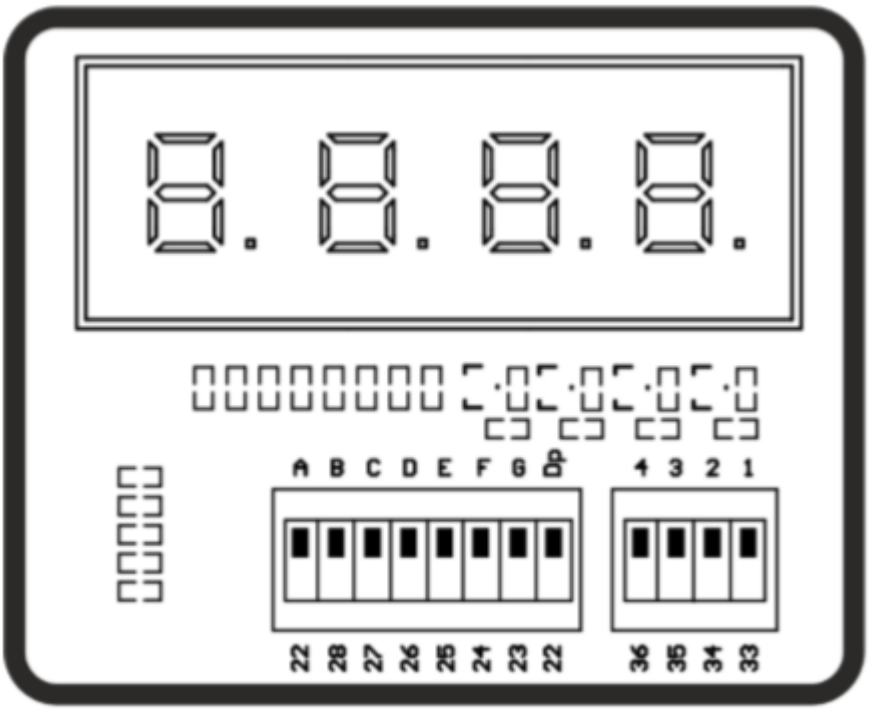
PRELIMINARY INFORMATION: In this experiment, we will give examples of more than one display control by using the previously used 7-segment display matrix method. In fact, if we use 4 displays, none will be active at the same time; we can run the desired displays in order and activate them for a short time so that the human eye will be deceived. This method is similar to the method of creating moving videos. Because the human eye can not perceive images that are repeated 20 times or more at the moment as separate photographs, movie frames are shot with 24 fps (frame per second) or when we look at the catalog information of web cams, we see a speed of 30 fps. When we increase the number of repetitions to more than 20 in this experiment, we perceive that the numbers are showed continuously in the displays.

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.
4- By making the necessary changes on the program, run the counter for it count between 0-999.

CIRCUIT DIAGRAM:





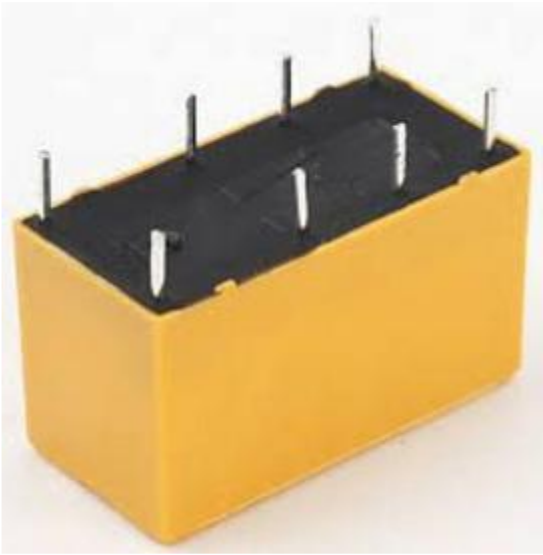
PROGRAM CODE:

```
int aPin = 22;      //          A
int bPin = 23;      //            _________
int cPin = 24;      //          |         |
int dPin = 25;      //     F    |         |    B
int ePin = 26;      //          |         |
int fPin = 27;      //          | _______ |
int gPin = 28;      //          |    G    |
int hpin = 29;      //          |         |
int GND1 = 33;      //          |         |
int GND2 = 34;      //     E    |         |    C
int GND3 = 35;      //          | _______ | o  DP(H)
int GND4 = 36;      //                D
      .
      .
      .
```

TEST NAME: RELAY CONTROL APPLICATION **TEST NO:7**

OBJECTIVE: To learn that LEDs are controlled as representations and that switching elements must be used for high power loads.

PRELIMINARY INFORMATION:

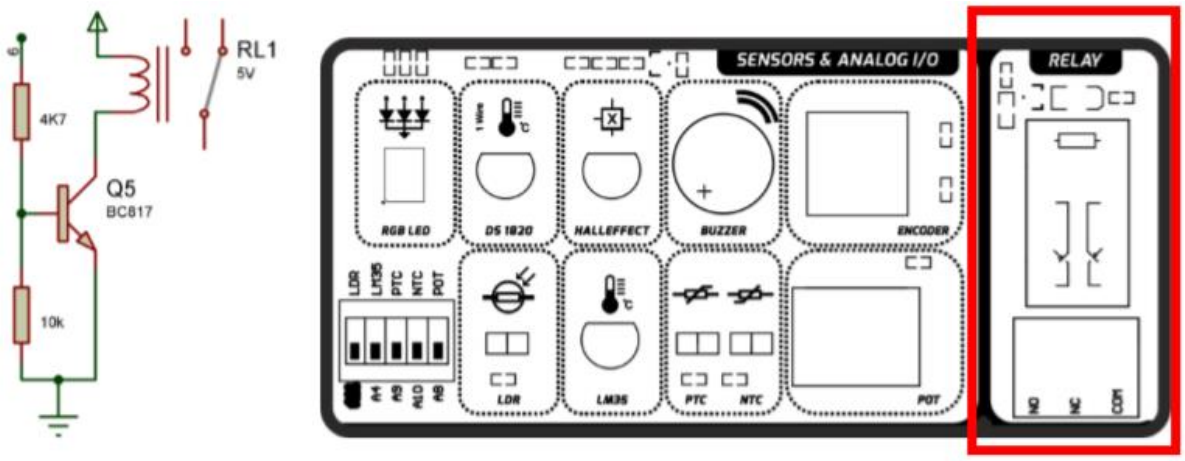


In practice, the maximum amount of current that microcontrollers can pass through is limited to 25mA. When working with higher currents, we need to use semiconductor or mechanical switching elements. In this experiment, we need to use both types of switching elements. The current required to operate the relay steadily is provided by the transistors. You can also visually check the status of the relay with the LEDs in front of the active relay.

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.
4- By making the necessary changes on the program, so that when button is depressed the relay is running, and when the button is released, the relay closes with 20 sec delay.

CIRCUIT DIAGRAM:



PROGRAM CODE:

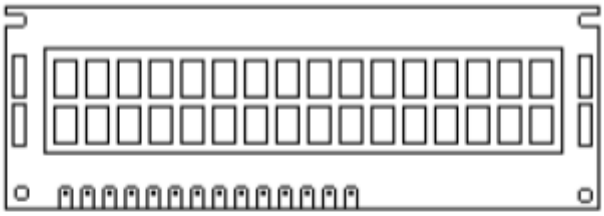
```
/*
        ARDUKİT &&&: RELAY CONTROL APPLICATION
        RELAY CONTROL WITH BUTTON;
        If the button is pressed which connected to pin number 38, therelay is run which connected with pin
        6.01/05/2016
*/
// Constants used in the program are defined in this section
          const int relay=6;
          const int button=38;
// Variables used in the program are defined in this section
          int i;
// Pre-settings are made with the setup function for the used pins and processor.
          void setup() {
// initialize digital pin 43-44-45-46 as an output.
                              pinMode(relay, OUTPUT);
                              pinMode(button, INPUT);
                    }
// Loop function is the section where we do the operations we want to run continuously.
void loop() {
                    if(digitalRead (button)==HIGH)
                    {
                              delay(20);
                              digitalWrite(relay,HIGH);
                    }
                    else
                  }           digitalWrite(relay,LOW);
```

TEST NAME: THE APPLICATION OF CHARACTER LCD **TEST NO: 8**

OBJECTIVE: By learning data types, perform character LCD application.

PRELIMINARY INFORMATION:

With Character LCD, we can show the characters in the ASCII code table, as well as the special characters we have created. There is 2x16 characters LCD in the experiment set, Which has 8 data lines and 3 control terminals, those data lines communicating with the microcontroller. The other terminals are the supply voltage leads for the LCD, and contrast adjustment and backlighting terminals. Backlighting terminals do not invisible at the table. In practice, we can make data communication over 4 bits with high weight. For we do not read from the LCD screen, we can use at the direct write mode of the R / W terminal (except in special cases). Those could used 0, and connect to the chassis level. A total of 32 characters can be displayed at the same time as 2 lines on the screen.

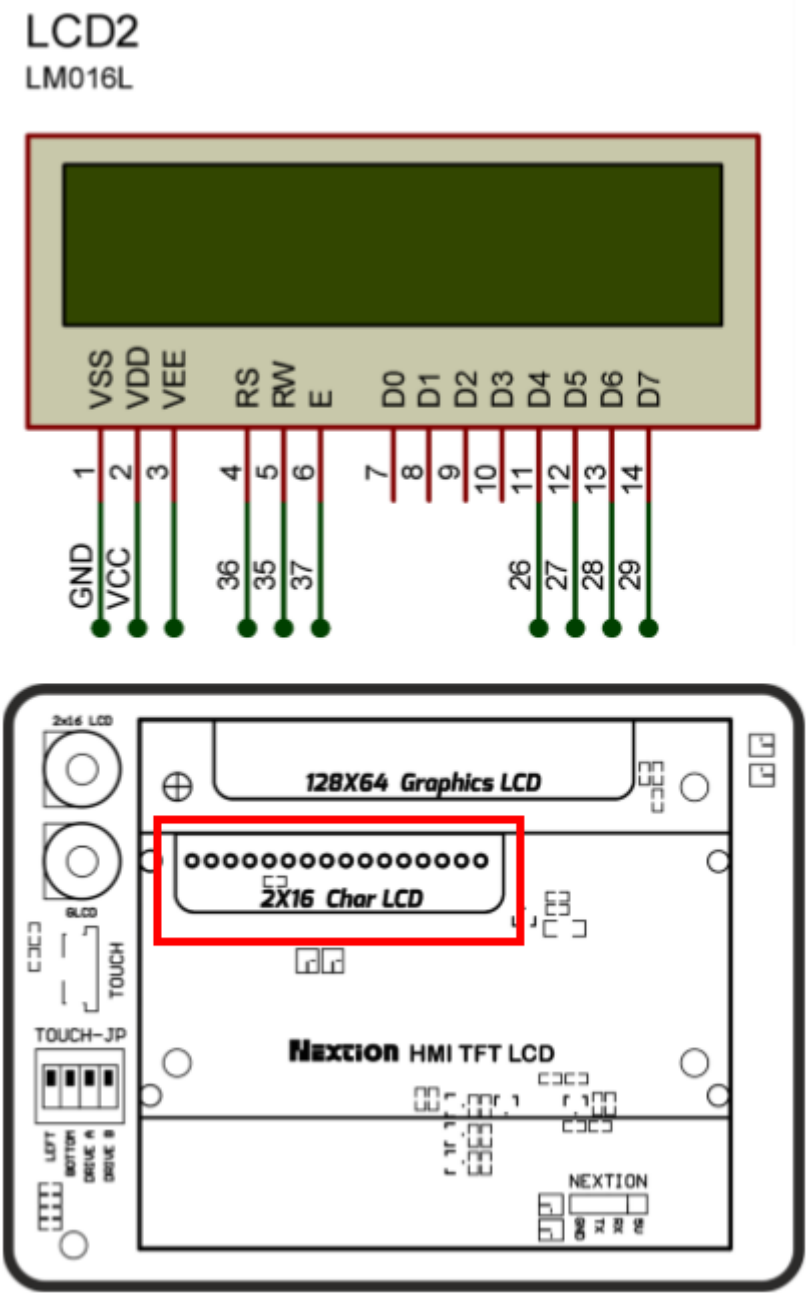


Pin No	Name	I/O	Description
1	Vss	Power	GND
2	Vdd	Power	+5v
3	Vo	Analog	Contrast Control
4	RS	Input	Register Select
5	R/W	Input	Read/Write
6	E	Input	Enable *(Strobe)*
7	D0	I/O	Data *LSB*
8	D1	I/O	Data
9	D2	I/O	Data
10	D3	I/O	Data
11	D4	I/O	Data
12	D5	I/O	Data
13	D6	I/O	Data
14	D7	I/O	Data *MSB*

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.
4- By making the necessary changes on the program, so that print your name and surname centered on the screen.

CIRCUIT DIAGRAM:



PROGRAM CODE:

```
/*
        ArduKIT &Character LCD Application
        LiquidCrystal Library - Inspired by Hello World
        Pin connections are the same as general examples
        Background lighting is connected to (backlight) pin 31 and
        RW controller is connected to pin 35.01/05/2016
        Library originally added 18 Apr 2008
        by David A. Mellis
        library modified 5 Jul 2009
        by Limor Fried (http://www.ladyada.net)
        example added 9 Jul 2009
        by Tom Igoe
        modified 22 Nov 2010
        by Tom Igoe
        This example code is in the public domain.
        http://www.arduino.cc/en/Tutorial/LiquidCrystal
*/
// The library file is added to the program in this section
        #include <LiquidCrystal.h>
//The pins to which the Character LCD monitor is connected
        LiquidCrystal lcd(36, 37, 26, 27, 28, 29);
        void setup() {
                        pinMode(31, OUTPUT);//BACKLIGHT
                        pinMode(35, OUTPUT);//RW
                        lcd.begin(16, 2);
// Writing message on LCD screenlcd.print(" GND KITS");
                        }
        void loop() {
// Screen is off
                        digitalWrite(31, LOW);
                        digitalWrite(35, LOW);
                        lcd.noDisplay();
                        delay(1500);
// Display is on
                        lcd.display();
                        digitalWrite(31, HIGH);
                        delay(1500);
}
```

TEST NAME: THE ADC APPLICATION WITH POTENTIOMETER **TEST NO: 9**

OBJECTIVE: To realize ADC application by learning the concept of Analog-Digital conversion.

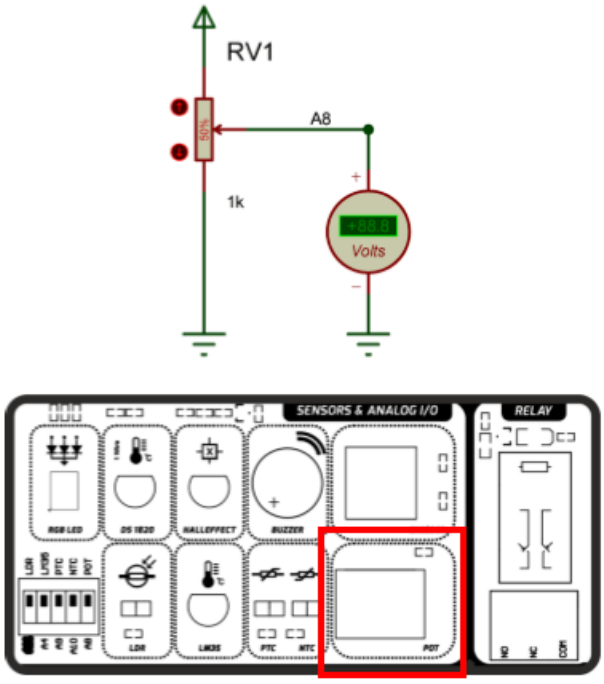
PRELIMINARY INFORMATION:

We encounter in daily life of all the physical quantities are all analog values. Analog-to-digital conversion is required to carry this data to digital machines. ADC (Analog Digital Converter) circuits are separated according to the cycle times and resolutions. For example, an 8-bit ADC circuit with a cycle time of 50 ms divides an analog value of 5 V into 256 parts and each part corresponds to 19.53125 mV. If the cycle end is 00011111, the measured voltage is approximately 0.6 V. If the cycle operation is 10 bits, the same voltage value is divided into 1024 pieces. The desired analog inputs are activated in the program by setup command. Potentiometer can be used for analog input operation also different sensors on the test set can be used for the same operation. In this experiment, the value read analog from the potentiometer will be displayed on the character LCD.

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.
4- Make the necessary changes on the program so that Print the analog value from 0 to 5 volts as it will be 1 digit after the comma (such as 3,3 V).

CIRCUIT DIAGRAM:



PROGRAM CODE:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(36, 37, 26, 27, 28, 29);
int pot=A8;
int analogdeger=0;
void setup() {    pinMode(31, OUTPUT);          //BACKLIGHT
                  pinMode(35, OUTPUT);          //RW
                  digitalWrite(31, HIGH);
                  digitalWrite(35, LOW);
                  lcd.begin(16, 2);              // Print a message to the LCD.
                  lcd.print("   GNDKITS");
               }
void loop() {     lcd.begin(16, 2);
                  lcd.print("DEGER= ");
                  analogdeger=analogRead(pot);
                  lcd.print(analogdeger);
                  delay(1000);
               }
```

TEST NAME: THE APPLICATION OF GRAPHIC LCD **TEST NO: 9**

OBJECTIVE: To learn logic of studying the Graphical LCD to perform basic drawing and writing operations.

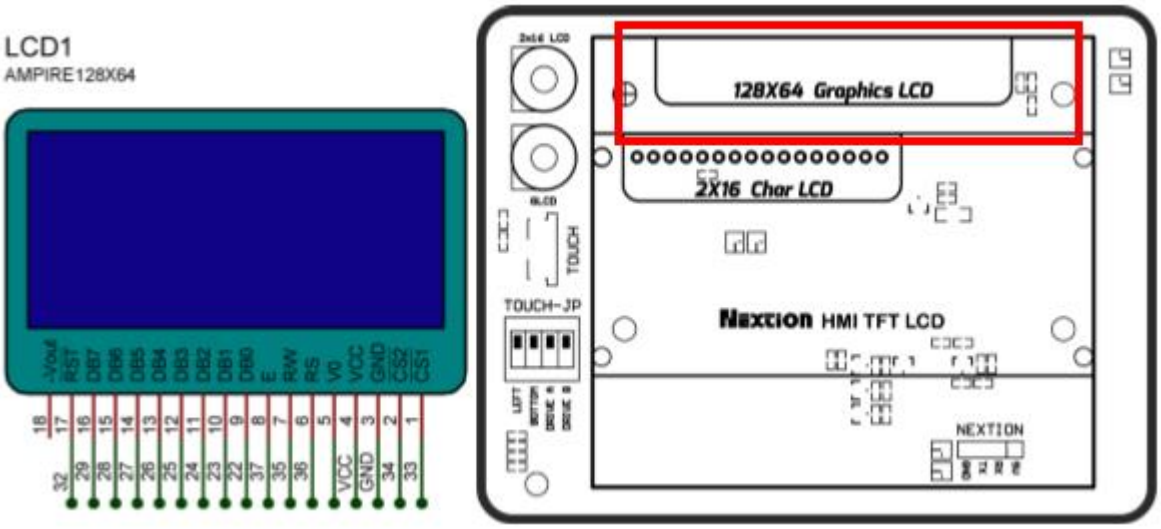
PRELIMINARY INFORMATION:

On the graphic LCD, individual access to the pixels is possible, unlike the character LCD. Many of the graphical LCD libraries prepared for Arduino are not compatible because they are prepared for different chipsets.

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.
4- Make the necessary changes on the program so that print your name, surname, your class and your number on the graphic LCD.

CIRCUIT DIAGRAM:



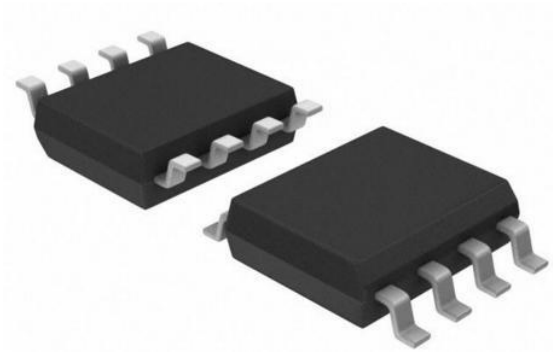
PROGRAM CODE:

```
/*      ArduKIT & The application of Graphic LCD
        * open GLCD Library - Inspired by Hello World      05/05/2016
        * 2013-06-15 bperrybap - updates for openGLCD
        * 2011-09-14 Bill Perry - original creation
        * bperrybap@opensource.billsworld.billandterrie.com
*/
#include <openGLCD.h>
void setup() {   pinMode(31, OUTPUT);              //BACKLIGHT
                 digitalWrite(31, HIGH);
                 pinMode(32, OUTPUT);              //RST
                 digitalWrite(32, LOW);
                 GLCD.Init();                      // GLCD preparing
                 GLCD.SelectFont(System5x7);       // Selecting font to use for writing
                 LCD.print("GNDKITS EXPERIMENT SETS ");
          }
        .
        .
```

TEST NAME: THE APPLICATION OF RTC **TEST NO: 11**

OBJECTIVE: Realizing the application by learning the consepts of I2C and Real Time Clock.

PRELIMINARY INFORMATION:

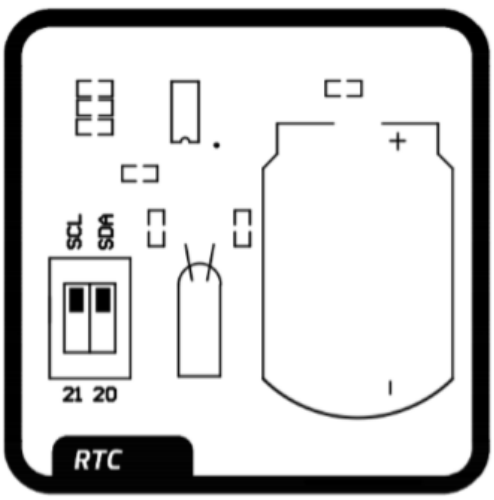


Digital machines have separate units that hold clock information in real time. The reason for this incident is the battery on the system. When requested, the time information can be read through this unit and combined with the desired data to keep the recordings in real time. In this experiment, the data of real time clock can also be monitored with the serial port “SERIAL MONITOR” while the graphic is printed on the LCD.

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.

CIRCUIT DIAGRAM:



Note: Don't forget to plug the battery into the test set

PROGRAM CODE:

```
/*      ArduKIT & RTC &The application of Graphic LCD
        Updates the RTC time depending on the system time.
        * open GLCD Library - Inspired by Hello World application and RTC library05/05/2016
        * 2013-06-15 bperrybap - updates for openGLCD
        * 2011-09-14 Bill Perry - original creation
        * bperrybap@opensource.billsworld.billandterrie.com
*/
// include the library header
// no font headers have to be included
#include <openGLCD.h>
#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 rtc;
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
                "Saturday"};
void setup() {  pinMode(31, OUTPUT);                    //BACKLIGHT
                .
                .
// This line sets the RTC with an explicit date & time, for example to set
// January 21, 2014 at 3am you would call:
// rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
        }
void loop() {   .
                .
                Serial.print('/');
                Serial.print(future.month(), DEC);
                Serial.print('/');
                Serial.print(future.day(), DEC);
                Serial.print(' ');
                Serial.print(future.hour(), DEC);,
                .
                .
```

TEST NAME: THE APPLICATION OF ROTARY ENCODER **TEST NO: 12**

OBJECTIVE: To learn the encoder's logic and apply.

PRELIMINARY INFORMATION:

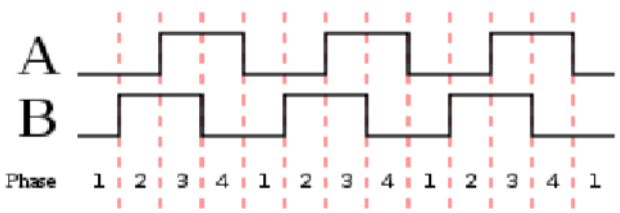
Rotary encoder has become a frequently used control tool in everyday life. This method, which is used to control motor rotation direction and speed in automation, has found a wide use possibility for control in electronic devices. It is working in Gray coding, one variable is changed at the each step. In this way, the velocity of the incoming pulses can be determined with respect to time, and the direction of rotation can be determined according to the order of the pulses.

Gray coding for clockwise rotation

Phase	A	B
1	1	0
2	1	1
3	0	1
4	0	0

Gray coding for clockwise rotation

Phase	A	B
1	1	0
2	1	1
3	0	1
4	0	0

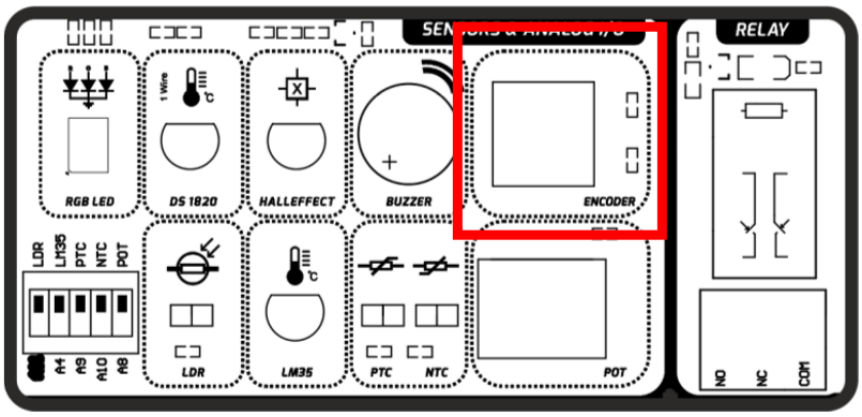


As you can see on the chart, only one of the phases A - B changes in each step. It compares the previous information with the new information in the program to determine turning position whether clockwise or counterclockwise.

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.

CIRCUIT DIAGRAM:



PROGRAM CODE:

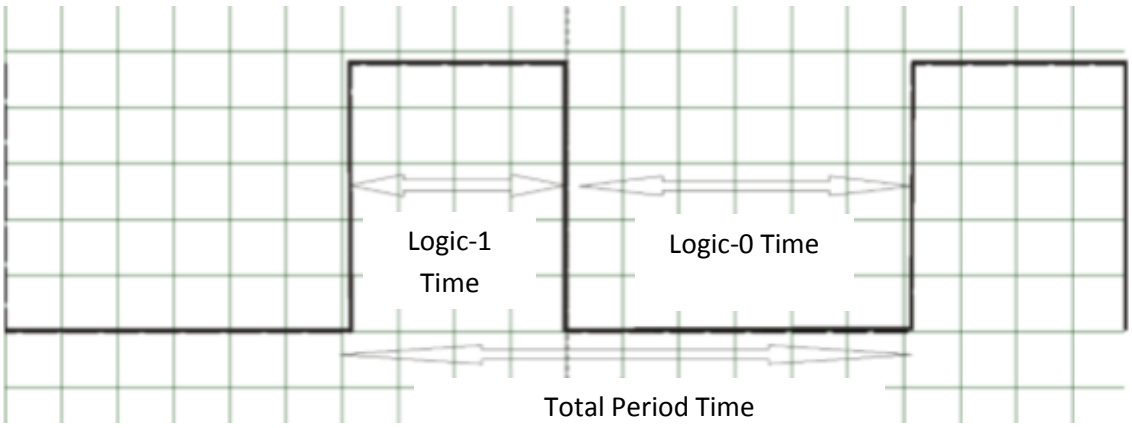
```
/*        Read Encoder
          * Connect Encoder to Pins encoder0PinA, encoder0PinB, and +5V.
          *
          * Sketch by max wolf / www.meso.net
          * v. 0.1 - very basic functions - mw 20061220
*/
int val;
int encoder0PinA = 48;
int encoder0PinB = 49;
int encoder0Pos = 0;
int encoder0PinALast = LOW;
int n = LOW;
void setup() {
                    pinMode (encoder0PinA,INPUT);
void loop() {
                    n = digitalRead(encoder0PinA);
                    if ((encoder0PinALast == LOW) && (n == HIGH)) {
                    if (digitalRead(encoder0PinB) == LOW)
                    {
                    encoder0Pos--;
                    }
          else {
                    encoder0Pos++;
                }
```

TEST NAME: THE CONTROL OF RGB LED WITH PWM **TEST NO:13**

OBJECTIVE: To learn consept of Pulse Width Modulation and apply.

PRELIMINARY INFORMATION:

Despite the presence of ADC units in the majority of microcontrollers we use, there is no DAC unit so that we can not produce Analog output. In these cases we use PWM (Pulse Width Modulation) technology to imitate analogue output. We can think of it simply as the average value calculated in AC current. If we accept the minimum value 0V, the maximum value 5V, the average value for a smooth square wave will be 2,5V. Here, in the total period time, if we increase of time 1 and decrease of time 0, the average value will increase, and if we decrease of time1 and increase of time 0, the average value will decrease.

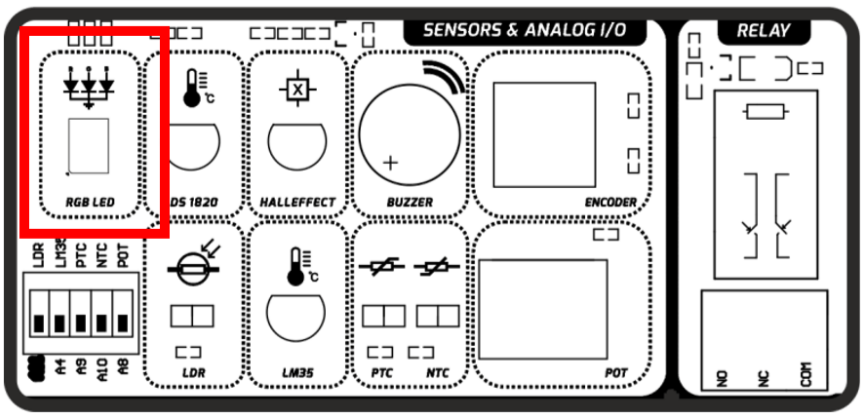


With the PWM method, many settings can be made, such as motor speed, RGB LED brightness. If the PWM signal that we use is obtained with CCP terminals, it takes the name of hardware pwm and works with specific pins. With timer interrupts we can get the name of the software pwm and use different pins for this purpose.

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.
4- Make the necessary changes on the program so that, make that all of the LEDs are lit when the first button is pressed and all of the LEDs are off when the second button is pressed.

CIRCUIT DIAGRAM:



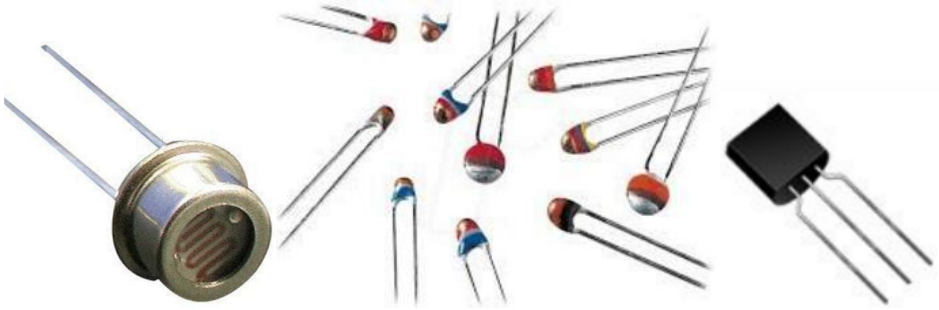
PROGRAM CODE:

```
/* Apply of ArduKIT & PWM RGB LED control and buzzer sound intensity are adjusted by PWM signal
   The value read from the potentiometer is shown on the character LCD. And this value is divided by 4
   and PWM value is generated between 0-255.
   According to the pressed button (38-39-40-41) the PWM value is diverted to a different analog output.
*/
// include the library code:
#include <LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
        LiquidCrystal lcd(36, 37, 26, 27, 28, 29);
        int pot=A8;
        int red=3;
        int green=4;
        int blue=2;
        .
```

TEST NAME: THE APPLICATION OF ANALOGUE SENSOR **TEST NO:14**

OBJECTIVE: Reading temperature data with the LM35 sensor and transfer to computer.

PRELIMINARY INFORMATION:



LDR **NTC-PTC** **LM35**

Analog quantities that we encounter in daily life are transmitted to the digital machines through sensors. We can briefly summarize the sensors on the test kit;

LDR: LDR (Light Dependent Resistors) It is a type of resistance that varies in value with light. As the light intensity increases, the resistance value decreases.

NTC: NTC (Negative Temperature Coefficient) It is a type of resistance that varies in value with tempareture. As the tempareture intensity increases, the resistance value decreases.

PTC: PTC (Positive Temperature Coefficient) It is a type of resistance that varies in value with tempareture. As the tempareture intensity increases, the resistance value increases.

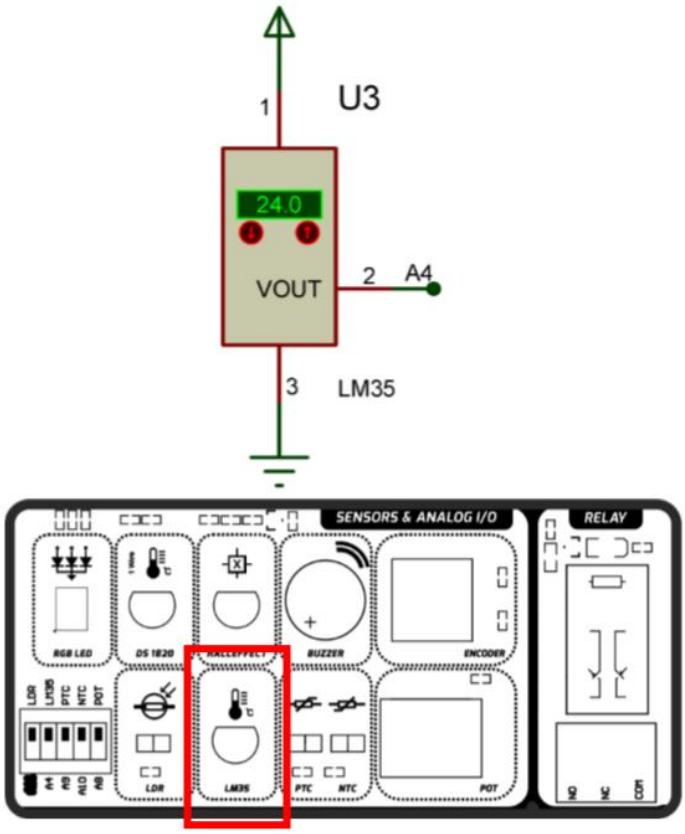
Since the first three sensors are variable resistors, a voltage divider circuit with fixed resistances of 4K7 is established and connected to the A9-A10 and A11 pins for measure the voltage change on the sensor. These sensors may have different values at room temperature; you must calibrate the program according to the type of sensor you are using.

LM35: It is a three-legged integrated circuit that is widely used for temperature measurement. The output voltage increases by 10 mV for every 1 degree rise in temperature. For example, it produces 0.25 V output voltage at 25 degrees celcius. The LM35 is connected to the A4 pin.

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.

CIRCUIT DIAGRAM:



PROGRAM CODE:

```
.
.
.
Serial.print("POTENTIOMETER =");Serial.println(analogvalue);
Serial3.print("POTENTIOMETER =");Serial3.println(analogvalue);
analogvalue =analogRead(lm35);
delay(20);
Serial.print("LM35 =");Serial.println(analogvalue);
Serial3.print("LM35 =");Serial3.println(analogvalue);
analogvalue =analogRead(ptc);
delay(20);
.
.
.
```

TEST NAME: THE APPLICATION OF 1 WIRE WITH DS18B20 **TEST NO: 15**

OBJECTIVE: To learn how to communicate over a single line by using the application of 1 WIRE with DS18B20.

PRELIMINARY INFORMATION:

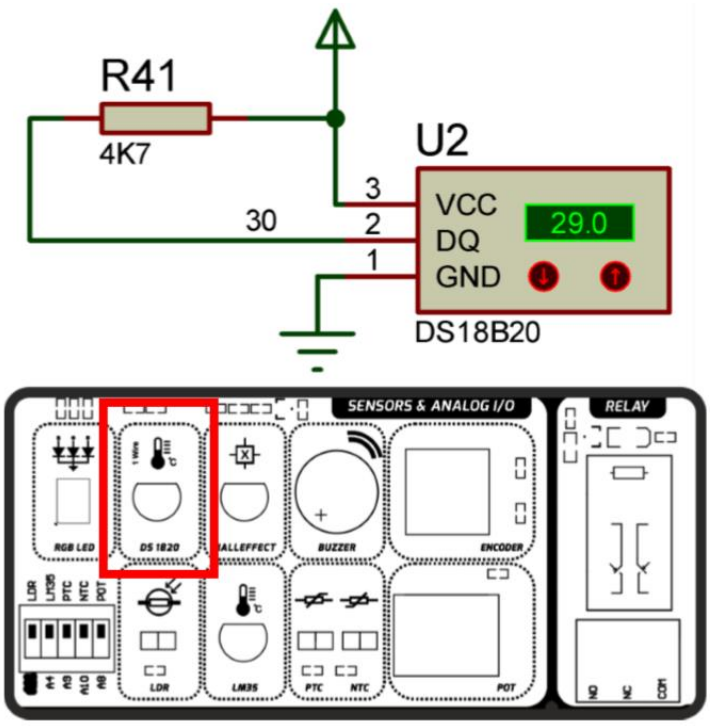
1 WIRE communication is a standard supported by many devices and sensors, from which you can collect information from more than one point over a single transmission line. For example, in a school with 24 classrooms, for to follow classroom temperatures from a single point, instead of connecting separate energy and data cables to each classroom, if you use only one cable line in classrooms and apply data and energy connections on that cable, you can read and understand temperature information from the each classroom. Each peripheral device has a unique ROM code of 64 bits (8 bytes). To read the ROM code, 0x33 code is sent when the sensor is attached alone, the first byte received at the time of communication contains the type of the peripheral device (0x28 for DS18B20), next 6 bytes contains of the serial number and the last byte contains of the CRC code. The ROM code of the peripheral device is the determinative and distinguishing feature of our computers like the Mach address, If the match code (0x55) + Rom code is sent after learning the ROM code of the peripheral device, only the temperature information from the relevant sensor is read. Most addressable sensors are connected with 1-wire communication. The DS 18B20 leg connections look like the following.

In this application you will measure with a single sensor using a driver file written for the DS18B20. If you wish, you can use multiple sensors with different driver files.

PROCESSING STEPS:

1- Install the program on ArduKIT by writing it with ArduinoIDE compiler or by drawing it with ArduBlock add-on package.
2- Perform the necessary connections on the sections shown in the connection diagram to provide the connections you see in the circuit diagram.
3- Observe the operation of the circuit.

CIRCUIT DIAGRAM:



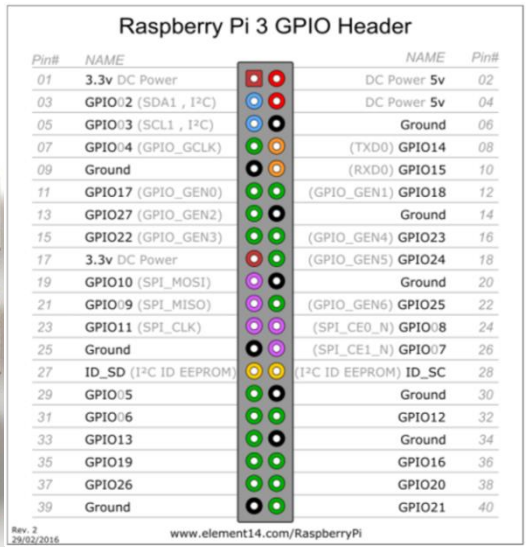
PROGRAM CODE:

```
.
.
void printTemperature(DeviceAddress deviceAddress)
{
        // method 1 - slower
        //Serial.print("Temp C: ");
        //Serial.print(sensors.getTempC(deviceAddress));
        //Serial.print(" Temp F: ");
        //Serial.print(sensors.getTempF(deviceAddress)); // Makes a second call to getTempC and then
        Serial.println(DallasTemperature::toFahrenheit(tempC)); // Converts tempC to Fahrenheit
        .
        .
```

TEST NAME: THE CONTROL OF THE LED WITH RASPBERRY **TEST NO:16**

OBJECTIVE: To learn how to control a physical output element with Raspberry Pi**.**

PRELIMINARY:



Raspberry Pi, which is a product of a project that started to spread programming education all over the world and enable children of low income countries to access a cheap computer. Now with Raspberry Pi-3 it fits all the features of an advanced computer on to a tiny card. Some basic applications can be made by connecting the 40 pin (B +) I / O port on the card to the ArduKIT training set. You can do your dreams when you use Arduino with Raspberry Pi or when you can program and run Arduino through Raspberry Pi. For example, with Raspberry Pi by combining the camera image processing application with a physical barrier control, you can perform chasing quickly the vehicle entry-exit with the plate.

PROCESSING STEPS:

1- Make connection between Raspberry Pi 40 pin port and ArduKIT.

2- Remember to make energy connection with adapter for Raspberry.

3- Run the program codes on the Raspberry Pi python interface, to observe that the system is running.

PROGRAM CODES:

The Lighting of One Led

To light the LED, the LED class from the GPiozero library is used. The required code, for to run:

```
from gpiozero import LED
led = LED(19)
led.on()
input()
```

Note: The last line is written to keep the program open until the enter key is pressed. Otherwise, the program will stop and the LED will turn off.

The Control of Multiple LEDs

Multiple LEDs can be identified similar to one LED.. Example code:

```
from gpiozero import LED
led1 = LED(19)
led1.on()
led2 = LED(20)
led2.on()
input()
```

Note: In the sample code, LEDs connected to pins 19 and 20 of Raspberry Pi were checked.

The Application of Flashing LED

The continuous flashing of the LED can be done by means of a handwritten code;

```
from gpiozero import LED
from time import sleep
led = LED(22)
while True:
led.on()
sleep(1)
led.off()
sleep(1)
```

Alternatively, blink can be used. Example code:

```
#-*-coding:utf-8-*-
from gpiozero import LED
led = LED(22)
led.blink(0.5, 0.5) # The time of On and Off.

input()
```

blink() It is written in turn, how many seconds are ON and how many seconds are OFF. In this example, the LED will light for half a second and turn off for half a second.

The Apply of Black Lightning

In this application, LEDs will be light respectively which is corresponding to Raspberry Pi's GPIO pins 19, 20, 21 and 22.

```
from gpiozero import LED
from time import sleep
led0 = LED(19)
led1 = LED(20)
led2 = LED(21)
led3 = LED(22)
while True:
led0.on()
sleep(0.5)
led0.off()
led1.on()
sleep(0.5)
led1.off()
led2.on()
sleep(0.5)
led2.off()
led3.on()
sleep(0.5)
led3.off()
```

TEST NAME: THE CONTROL OF BUTTON WITH RASPBERRY PI **DENEY NO:17**

OBJECTIVE: To learn how to control a physical input element with Raspberry Pi.

PRELIMINARY INFORMATION:

As an example of the input elements that can be used physically, the input operation with the button is performed in this application. RPi.GPIO is used as an example for button control. In pin numbering process, the pin numbering of the card on Raspberry Pi was followed, unlike in gpiozero. You can look at pinout.xyz for more information.

PROCESSING STEPS:

1- Make connection between Raspberry Pi 40 pin port and ArduKIT.

2- Remember to make energy connection with adapter for Raspberry.

3- Run the program codes on the Raspberry Pi Python interface, to observe that the system is running.

PROGRAM CODE:

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(5, GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(40, GPIO.OUT)
try:
while True:
        if GPIO.input(5):
            print " Port 3 logic 1 - button pressed "

                GPIO.output(40, False)
        else:
        print " Port 3 logic 0 - button not pressed "
        GPIO.output(40, True)
        sleep(0.1)
finally:
GPIO.cleanup()
```

In the code, the pin of number 5 GPIO pin (card numbering) is defined as input. The LED to be used to recognize that the button is pressed is also designated as the LED connected to the GPIO 40 in the same numbering. The pin to which the LED is connected is set as output in this case.

On the last line of the program, the cleanup () function is called so that the next time the RPi.GPIO library is run, it will not give a relevant warning that the pins are already set.

TEST NAME: THE CONTROL OF 2x16 LCD WITH RASPBERRY PI **TEST NO: 18**

OBJECTIVE: To perform character LCD usage with Raspberry Pi.

PRELIMINARY INFORMATION:

The library must be loaded before the LCD can be used. If the LCD is used alone, it must be loaded and then adjusted in the code. The following command should be written to the terminal for installation:
wget https://bitbucket.org/MattHawkinsUK/rpispy-misc/raw/master/python/lcd_16x2.py

The required settings of the code are as follows. (This code needs to be saved as lcd.py):

```
#!/usr/bin/python
#
# HD44780 LCD Test Script for
# Raspberry Pi
#
# Author : Matt Hawkins
# Site : http://www.raspberrypi-spy.co.uk
#
# Date : 03/08/2012
#
# The wiring for the LCD is as follows:
# 1 : GND
# 2 : 5V
# 3 : Contrast (0-5V)*
# 4 : RS (Register Select)
# 5 : R/W (Read Write) - GROUND THIS PIN
# 6 : Enable or Strobe
# 7 : Data Bit 0 - NOT USED
# 8 : Data Bit 1 - NOT USED
# 9 : Data Bit 2 - NOT USED
# 10: Data Bit 3 - NOT USED
```

```
    GPIO.output(LCD_D7, True)
    # Toggle 'Enable' pin
    time.sleep(E_DELAY)
    GPIO.output(LCD_E, True)
    time.sleep(E_PULSE)
    GPIO.output(LCD_E, False)
    time.sleep(E_DELAY)
if __name__ == '__main__':
main()
```

** You can find the relevant file (lcd.py) in the ARDKIT experiment folder.

PROCESSING STEPS:

1- Make connection between Raspberry Pi 40 pin port and ArduKIT.

2- Remember to make energy connection with adapter for Raspberry.

3- Run the program codes on the Raspberry Pi python interface, to observe that the system is running.

PROGRAM CODE:

```
import RPi.GPIO as GPIO ## Import GPIO Library
import sys
import lcd
import time
GPIO.setmode(GPIO.BOARD) ## Use BOARD pin numbering
GPIO.setup(22, GPIO.OUT) ## Setup GPIO pin 7 to OUT
GPIO.setup(40, GPIO.OUT) ## Setup GPIO pin 7 to OUT
GPIO.output(40, False) ## Turn on GPIO pin 7
lcd.lcd_init()
##GPIO.output(22, True) ## Turn on GPIO pin 7
lcd.lcd_byte(lcd.LCD_LINE_1, lcd.LCD_CMD)
lcd.lcd_string("Raspberry Pi", 2)
lcd.lcd_byte(lcd.LCD_LINE_2, lcd.LCD_CMD)
lcd.lcd_string("Model B+", 2)
time.sleep(10)
lcd.GPIO.cleanup()
```

TEST NAME: ARDUINO USB CONNECTION TEST WITH RASPBERRY PI **TEST NO:19**

OBJECTIVE: In the Raspberry Pi environment, to be follow up the data sent in series with Arduino.

PRELIMINARY INFORMATION:

In this section, you will be able to see how the data sent with Arduino's serial port can be seen on Raspberry Pi. Make sure the Arduino is plugged into the Raspberry Pi via the USB port. So that, the USB-UART converter Raspberry Pi in the Arduino part is connected to this and the information that Arduino sends from UART can be transferred to Raspberry Pi. The USB port is visible on the system with / dev / ttyUSB0 or similar names.

PROCESSING STEPS:

1- Make connection between Raspberry Pi 40 pin port and ArduKIT.

2- Remember to make energy connection with adapter for Raspberry.

3- Run the program codes on the Raspberry Pi python interface, to observe that the system is running.

PROGRAM CODE:

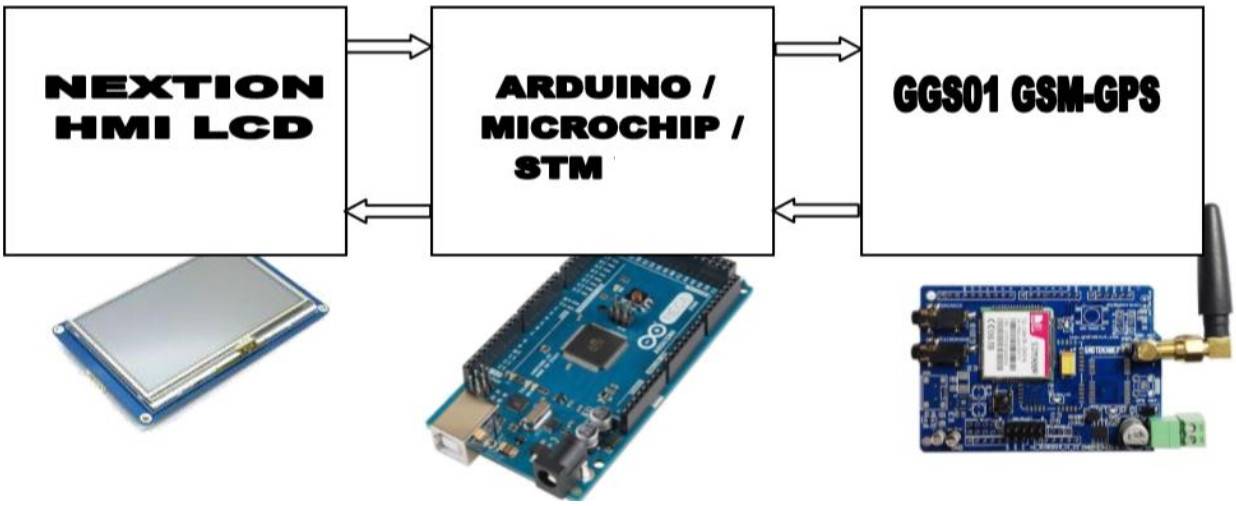
```
import serial
import time
ser = serial.Serial('/dev/ttyUSB0')
data = ser.readline()[:-2]
print data
If more than one piece of data comes in, separated by a comma, it can be parsed as follows:
import serial
import time
ser = serial.Serial('/dev/ttyUSB0')
data = ser.readline()[:-2]
datas = data.split(“,”)
print datas
In order to be able to read continuously, the following usage will be suitable:
import serial
import time
ser = serial.Serial('/dev/ttyUSB0')
while True:
data = ser.readline()[:-2]
datas = data.split(“,”)
print datas
```

TEST NAME: THE APPLY OF GND PHONE **TEST NO:20**

OBJECTIVE: Performing a sample operation for multi-serial applications.

PRELIMINARY INFORMATION:

With this example, a small phone application was realized by using Nextion Hmi screen which is known as intelligent TFT screen which can communicate over serial port. Unlike the previous applications, the GGS01 GSM / GPS module, HMI display and ArduKIT are used at the same time and messages with different content can be sent to the desired number. With the TX3-RX3 pins located on the Arduino mega, search-messaging-querying of GPS coordinates can be performed both through the Arduino IDE serial port screen and via the touch screen. If you can not run this application because it is more advanced than the previous examples, you should first run other examples and understand the use of GSM and GPS modules of GGS01 product.



PROCESSING STEPS:

1- Place the GGS01 module on the ARDKIT, paying attention to the reference holes.

2- For detailed information on the jumper settings of the GGS01 module, please read the operating instructions.

3- Install the "gndphone.hmi" file in the Nextion-Hmi screen and the "phone.ino" file in the arduino, by performing the connections. You can find detailed information about using the Nextion-HMI display at http://wiki.iteadstudio.com/Nextion_HMI_Solution.

4- Detailed information about the Arduino program can be found in the introduction to the program code. You can find the complete code in the corresponding experiment folder.

PROGRAM CODE:

```
/*
GND TECHNIC
A SAMPLE APPLICATION OF WITH NEXTION HMI DISPLAY AND GGS01 MODULE
ANY PHONE NUMBER CAN BE CALL & MESSAGE SEND ON THE COMPUTER OR HMI DISPLAY.
FOR CALL OPERATION a 5XXXXXXXXX ENTER (10 13)
TO SEND SMS m 5XXXXXXXXX MESSAGE ENTER (10 13)
FOR GPS DATA SEND g ENTER (10 13)
DATA FROM GPS MODULE THAT START WITH $GPRMC IS SELECTED AND PRINT ON THE SCREEN
FOR OTHER CODE SAMPLES SIM28 DATASHEET CAN BE CHECK
Table 2-10: RMC Data Format
Example: $GPRMC,094330.000,A,3113.3156,N,12121.2686,E,0.51,193.93,171210,,,A*68<CR><LF>
Name Example Unit Description
Message ID $GPRMC RMC protocol header
UTC Time 094330.000 hhmmss.sss
Status [1] A A=data valid or V=data not valid
Latitude 3113.3156 ddmm.mmmm
N/S Indicator N N=north or S=south
Longitude 12121.2686 dddmm.mmmm
E/W Indicator E E=east or W=west
Speed Over Ground 0.51 knots
Course Over Ground 193.93 degrees True
Date 171210 ddmmyy
Magnetic Variation [2] degrees E=east or W=west
East/West Indicator[2] E=east
Mode A A=Autonomous,
D=DGPS
Checksum *68
<CR><LF> End of message termination
```

ARDUINO MEGA USED BECAUSE IT INCLUDES 4 USART UNITS

USART 0 - FOR COMMUNICATION WITH PC

USART 1- FOR COMMUNICATION WITH GSM MODULE

USART 2 - FOR COMMUNICATION WITH GPS MODULE

USART 3 – FOR COMMUNICATION WITH HMI DISPLAY

AIM OF USE THIS APPLICATION TO PROVIDE ENVIRONMENTAL DEVELOPMENT FOR EDUCATION PURPOSE

SO CODES ARE NOT OPTIMIZED.

LONGEST METHODS USED FOR SOME OPERATIONS.

YOU CAN SHARE UPDATES YOU WANT YOUR DIVERSITY IN YOUR APPLICATION.

ArduKIT

ARDUINO & RASPBERRY PI DEVELOPMENT KIT

www.gndteknik.com

GNDTEKNİK

GND TEKNİK EDUCATION TECNOLOGIES
ELECTRONIC AUTOMATION SYSTEMS
Adresses :
Uzuncayır Street No: 30 Konak Business Center
Floor: 3 Office: 55 Hasanpasa Kadikoy
Tel *: +90216 428 66 55*
Fax *: +90850 225 25 45*
E-mail : *info@gndteknik.com*