Tinylab

Kerem İZGÖL
Yasir ÇİÇEK

abaküs

abaküs 073
Tinylab
Kerem İZGÖL - Yasir ÇİÇEK
1. Baskı: Mayıs 2017
ISBN: 978-605-9129-73-2

Kapak ve sayfa düzeni: Cem Demirezen
Düzelti: Özcan Oğuz
Satış: satis@abakuskitap.com
Baskı: Ezgi Matbaacılık San Tic. Ltd. Şti.
Sanayi Cad. Altay Sok. No:14
Çobançeşme-Yenibosna/İSTANBUL
Tel: 0212 452 23 02
Matbaa Sertifika No: 12142

Selda Ustabaş Demiryakan
Abaküs Kitap Yayın Dağıtım Hizmetleri
Yayıncılık sertifika no: 31359
Hobyar Mah. Cemal Nadir Sok. No:24/227 Cağaloğlu-Fatih/İST.
Tel.: (0212) 514 68 61
www.abakuskitap.com - editor@abakuskitap.com

Kerem IZGOL

I was born in Beşiktaş on October 30, 1991. Since my childhood, I have had a passion and curiosity for electronic devices, which led me to choose Control and Automation Engineering as my profession. I believe that no knowledge is unnecessary, so I constantly strive to learn new things.

Having a maker spirit, I prefer to create things myself as much as possible rather than buying ready-made products, and I enjoy the process of learning new things through research. Additionally, as a maker, I aim to teach what I learn as accurately as possible. Remember, sharing knowledge leads to its expansion.

I have written this book to introduce you to the world of Arduino with a platform like Tinylab to the best of my ability and knowledge. I hope it serves as a valuable resource for you. Don't be afraid to create!

Yasir CİCEK

I was born in Karabük on November 24, 1990. I am an Electronics and Telecommunications Engineer and a volunteer in the maker movement. My journey in education started at an early age due to my father being a teacher. I can say that I have always been closely involved with education and school throughout my life.

Having attended a teacher training high school, education became an indispensable field for me. I have been involved in educational activities within Robotistan for a long time. I love education, teaching, learning new things, and sharing knowledge. I hope that this book we have written will serve as a valuable resource for those who want to learn and teach Arduino and Tinylab..

Thank you

I would like to express my gratitude to the Robotistan family for providing me with the opportunity to write this book, and to my esteemed mentor Dilek Bilgin Tükel, who introduced me to Arduino.

Kerem Izgol

I would like to extend my thanks to my fellow author Kerem İzgöl and the Robotistan family for their unwavering support during the preparation of this book. I am also grateful to my spouse and family for their encouragement.

Yasir Cicek İstanbul, 2016

İÇİNDEKİLER

1. Tanıtım 1

Bu Kitap Kimler için? 1

Maker Nedir? 2

Arduino'nun Hikâyesi 4

Klon Kartlar 7

Tinylab 8

Arduino IDE ve Sürücülerin Kurulumu 9

Kütüphaneler 12

Windows için 13

Mac OS için 14

GNU/Linux için (Ubuntu MATE 16.04 LTS kullanılmıştır) 15

2. Dersler 17

Ders 01: LED Yakıp Söndürmek (Blink) 18

LED Nedir? 18

Gerilim, Akım ve Ohm Yasası 18

Ders 02: RGB LED Kontrolü 22

PWM ve Dijitalden Analog'a Dönüşüm 23

Ders 03: Seri Haberleşme 29

Ders 04: Dijital Girişler 34

Ders 05: Analog Girişler 37

Analogtan Dijitale Çeviriciler

(Analog-to-Digital Converter, ADC) 37

Potansiyometre 38
Light Dependent Resistor (LDR, Foto Direnç) 38
Sıcaklık Sensörü 39
Ders 06: Buton Kontrollü RGB LED 42
Ders 07: Buzzer ile Ses Çıkışı 49
Notalar ve Frekans 49
Ders 08: 16x2 LCD Ekran 52
Ders 09: 7-Parçalı LED Gösterge (7-Segment Display) 57
Ders 10: Rotary Encoder 64
Ders 11: DC Motor Hız Kontrolü 69
Ders 12: Gerçek Zamanlı Saat (RTC) 76
Ders 13: Harici EEPROM 80
Ders 14: SD Kart Kullanımı 85
3. Projeler **93**
7-parçalı Ekran Motor Hız Göstergesi 94
ESP8266 WiFi Modülü Kullanımı 100
ESP8266 ile ThingSpeak'e Sensör Bilgisi Gönderme 104
7-segment ile Rotary encoder 110
Delay'siz Arduino Kullanımı 116
Termometre ve Takvimli Saat 120
RC522 RFID Modülü ile Röle Kontrolü 127
HC06 Bluetooth Modülü ile Akıllı Cihaz Üzerinden LED Kontrolü 134

XBee Uzak Sensör Modülü 142
MPU6050 6-eksenli İvmeölçer ve Jiroskop Kartı 146 nRF24L01 ile Kablosuz Gamepad 157

Imagine, Design, Create

As a student team with excitement for technology innovation and ambitious dreams, we embarked on our journey in 2010. By selling our produced boards and robot kits, we opened our e-commerce website with the dream of establishing an R&D budget for ourselves... We still laugh whenever optimistic scenarios come to mind :)

Actually, our dreams were quite realistic, and we can say they slowly turned into reality. We just didn't know it would be such a long and challenging road. Throughout this process, as a company founded with a student budget, we learned how difficult it is to conduct clean trade and R&D within ethical boundaries in our country's conditions. While gaining these experiences, we realized something invaluable, even more valuable

than our greatest excitement of producing: teaching production. We must break the cycle of consumer culture growing like an avalanche and remind people of the joy of production.

We see production as the only way out to solve our country's problems. But even more importantly, we believe we cannot afford the luxury of leaving the problems of our world, to generations lacking problem-solving abilities, ideas, art, technology, and solutions. As the Robotistan team, we will continue to work with all our might to raise generations in our country who love to produce.

#Don’tBeAfraidToProduce

Robotistan

1

Introduction

Who Is This Book For?

This book appeals to a wide range of audiences, including students, engineers, teachers, and individuals of all ages and levels of knowledge/skill who are interested in learning Arduino programming.

Today, various starter kits are being prepared for those who want to learn Arduino. These sets offer the essential components needed for individuals to step into the world of microcontroller programming as a ready-to-use package. Learning Arduino programming and engaging in various projects in this field requires some skills in electronic circuit assembly. While this may be a straightforward step for some individuals, at times, setting up the circuit can prolong the learning process by involving inevitable trial-and-error and troubleshooting difficulties, which can make the process less enjoyable.

TinyLab is a product developed to facilitate this process. Components come pre-installed on the board, allowing you to save time by skipping potential problems during circuit assembly and directly moving on to coding.

Not only beginners but also experienced individuals can benefit from the practical usage of TinyLab. By encompassing commonly used

components in microcontroller projects, it accelerates your prototyping process significantly.

In order to appeal to not only TinyLab users but also anyone interested in learning Arduino, each lesson and project in the book includes a list of materials and schematics for circuits that you can build yourself.

The lessons section of the book is designed to easily teach the usage of each component found on TinyLab through easily understandable examples, making it suitable for those with little experience or knowledge. Starting from the most basic microcontroller applications and progressing towards more complex ones, each application revisits the topics covered in the previous lessons to build a strong foundation. As a result, you will be able to easily tackle the projects in the second part of the book. After completing the book, the only obstacle in creating your own projects will be the limits of your imagination.

What Is Maker?

A maker is someone who, in the simplest definition, creates and produces things. The culture known as DIY (Do it yourself) also falls under the maker umbrella. Within this scope, individuals who work on projects using Arduino and similar electronic hardware, design and print things with 3D printers, build their own drones, and even those who enjoy cooking and derive pleasure from it fit the maker definition.

The essence of the maker culture is primarily nourished by sharing and experiences. A maker, regardless of what they create, prefers to share rather than keep to themselves. This includes sharing knowledge and experiences. For makers, experience holds more importance than rote memorization. They understand that memorized knowledge may not play as effective a role as experience in creating something. The process of gaining experience typically follows this pattern: Do, fail, and do again. Therefore, a maker is someone who is not afraid of failure.

Today, with the widespread availability of technology and its accessibility to everyone, creating anything has become much easier. Thanks to the internet, access to all kinds of resources is possible. Therefore, detailed explanations, circuit diagrams, 3D printer model files, and other materials related to what you want to create can easily be obtained.

Since sharing is at the heart of the maker culture, there are spaces such as makerspaces or coworking spaces where makers can work collectively. In these workshops, people of all experience levels can come together to share their dreams, experiences, and ideas. Especially, such spaces organize educational events tailored for beginners. If you have the maker spirit in you and want to start somewhere, you can follow the activities of these communities.

One of these spaces is Makerhane. Located in Kadıköy, Istanbul, Makerhane offers a warm cafe atmosphere where you can work on your projects in its workshop and participate in various events and workshops. Additionally, you can easily obtain materials for your project from the market section, benefit from 3D printing services, use the free maker library, and even organize your own event at Makerhane if you wish..

The Story of Arduino

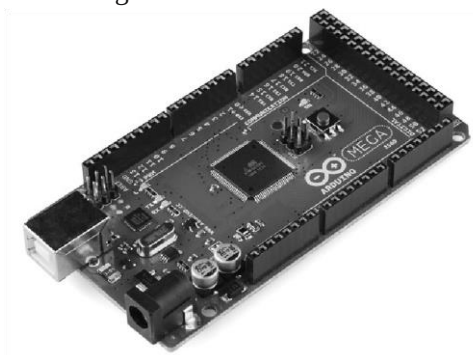
- Arduino is a microcontroller platform that was developed in Italy in 2004. Its primary goal during design was to be user-friendly, unlike other microcontroller platforms, and to appeal not only to technical experts like engineers but also to everyone, making it easily accessible. As a result, unlike other microcontroller platforms, it garnered significant interest from various communities and quickly became popular.

- The Arduino platform primarily consists of microcontrollers from Atmel's AVR series. It might be necessary to briefly explain what a microcontroller is. A microcontroller is a single integrated circuit

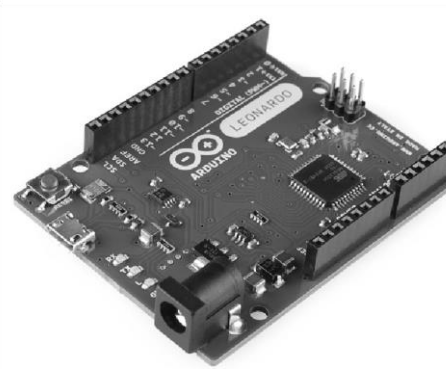
(chip) that comprises a processor, memory, and programmable input/output units, essentially constituting a mini-computer. In other words, the code we write is loaded into the memory inside this integrated circuit, starts running when powered, and controls the input/output connection pins on the integrated circuit according to the program we wrote. There are different Arduino models based on different microcontrollers and sizes. Some of these models include:

- **Arduino Uno:** The most well-known and widely used Arduino model is Arduino Uno. Thanks to its built-in USB port, it can be easily connected to a computer for uploading code. It features 14 digital input/output pins (6 of which can be used as PWM outputs), 6 analog inputs, 32 KB of flash program memory, 2 KB of SRAM memory, and 1 KB of built-in EEPROM memory. It was initially released as the Arduino USB model and evolved over time, being successively named Arduino Extreme, Arduino NG, Arduino Diecimila, Arduino Duemilanove, and finally Arduino Uno. Models up to Duemilanove feature an 8-bit ATmega168 microcontroller, while the current models have been upgraded to the ATmega328 microcontroller.

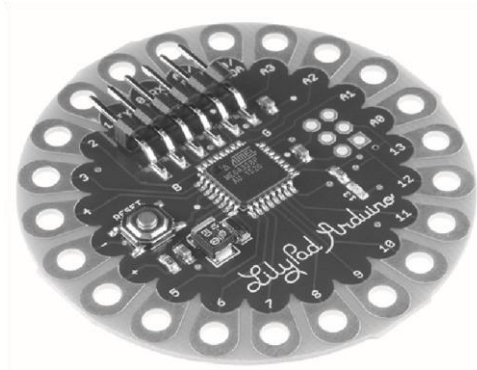
- **Arduino Mega 2560:** Arduino Mega, developed for use in projects where Arduino Uno's number of input/output pins and memory are insufficient, features 54 digital input/output pins (15 of which can be used as PWM outputs), 16 analog inputs, 256 KB of flash program memory, 8 KB of SRAM memory, and 4 KB of EEPROM memory. Previous models are equipped with the ATmega1280 microcontroller and are simply referred to as Arduino Mega. The current version, however, utilizes the ATmega2560 microcontroller, hence the name Mega 2560.



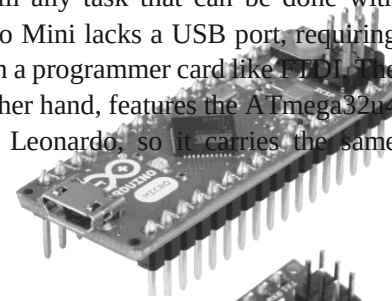
- **Arduino Leonardo:** Our TinyLab board is based on the Arduino Leonardo. Unlike the Uno, this Arduino model shares the same board design but features the ATmega32u4 microcontroller. The most significant difference is that this microcontroller has built-in USB connectivity and can be recognized by the connected computer as peripheral devices such as a keyboard or mouse. Additionally, since communication with the computer and the external serial port are separate, it does not interfere with

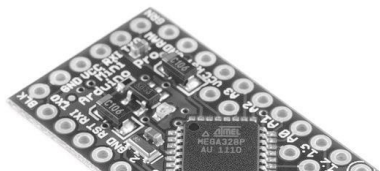


- modules that use UART communication, such as XBee, Bluetooth modules, or ESP8266, providing great convenience. It has 20 digital input/output pins (7 of which can be used as PWM outputs), 12 analog inputs, 32KB of program memory, 2.5 KB of SRAM, and 1 KB of EEPROM memory.
- **LilyPad Arduino Main Board:** LilyPad is one of the Arduino boards designed to have a sewable structure. It has a round shape, and its connections are arranged around the board's perimeter for easy sewing onto garments. With its ATmega328 processor, it can perform almost any task that the Uno model can. In the version of LilyPad with the ATmega328 microcontroller, there is no USB connection, requiring the use of a separate FTDI or similar USB-to-serial converter and programmer for programming purposes.



- **Arduino Pro Mini, Micro ve Nano:** Arduino Pro Mini and Nano are small-sized models of Arduino boards designed specifically for integration into projects where space is limited. Additionally, their pin layouts allow them to be easily attached to breadboards. Both Arduino Pro Mini and Nano versions feature the ATmega328 microcontroller and can perform any task that can be done with Arduino Uno. Only Arduino Pro Mini lacks a USB port, requiring software uploads to be done with a programmer card like FTDI. The Arduino Micro model, on the other hand, features the ATmega32u4 microcontroller, similar to the Leonardo, so it carries the same features.





- **Genuino 101:** The Genuino 101, which features an Intel Curie 32-bit processor, offers high processing and memory capabilities, Bluetooth Low Energy (BLE) support, and accelerometer functionality, making it highly useful. The board's size and pin layout are the same as Arduino Uno's.

	Mikrokontrolcü	Dijital Giriş/Çıkış Sayısı	Analog Giriş Sayısı	Bellek	Bağlantı Tipi	Kullanım Alanı
Arduino UNO	ATmega328p, 16 MHz, 16-bit	14 (6 adet PWM çıkış)	6	1KB EEPROM, 2KB SRAM, 32KB Flash	USB-B	Giriş seviyesi temel kullanım.
Arduino MEGA 2560	ATmega2560, 16 MHz, 16-bit	54 (15 adet PWM çıkış)	16	4KB EEPROM, 8KB SRAM, 256KB Flash	USB-B	Ekstra bellek ve g/ç sayısına ihtiyaç duyulan projeler.
Arduino Leonardo	ATmega32u4, 16 MHz, 16-bit	20 (7 adet PWM çıkış)	12	1KB EEPROM, 2,5KB SRAM, 32KB Flash	USB Mikro-B	Giriş seviyesi temel kullanım. Ekstra özellikler sunar.
LilyPad Arduino Main Board	ATmega328p, 8 MHz, 16-bit	14 (6 adet PWM çıkış)	6	512 Byte EEPROM, 1KB SRAM, 16KB Flash	FTDI Konektörü	Giyilebilir projeler.
Arduino Mini	ATmega328p, 16 MHz, 16-bit	14 (6 adet PWM çıkış)	8	1KB EEPROM, 2KB SRAM, 32KB Flash	FTDI Konektörü	Gömülü projeler içerisinde kullanım.
Arduino Micro	ATmega32u4, 16 MHz, 16-bit	20 (7 adet PWM çıkış)	12	1KB EEPROM, 2,5KB SRAM, 32KB Flash	USB Mikro-B	Leonardo ile aynı özellikler, proje içine gömülmeye uygun boyut.
Arduino Nano	ATmega328p, 16 MHz, 16-bit	14 (6 adet PWM çıkış)	8	1KB EEPROM, 2KB SRAM, 32KB Flash	USB Mini-B	UNO ile aynı özellikler, proje içine gömülmeye uygun boyut.
Genuino 101	Intel Curie, 32 MHz, 32-bit	14 dijital g/ç (4 adet PWM çıkış)	6	24 KB SRAM, 196 KB Flash	USB-B	32-bit Intel Curie platformuna giriş amaçlı, BLE ve ivmeölçer gibi ek özellikler.

Clone Boards

Since the Arduino platform is entirely open-source (both in terms of hardware and software), all circuit diagrams, program codes, and PCB layouts are readily accessible. Consequently, you can produce your own Arduino board. The easy accessibility of these resources has also enabled different manufacturers, particularly in countries with low production costs like China, to produce Arduino boards. These types of boards are commonly referred to as "clones" in the market and typically come with

- more affordable price tags. While most clone boards offer the same functionality as the original, the only potential issue could be the manufacturing quality.

Tinylab

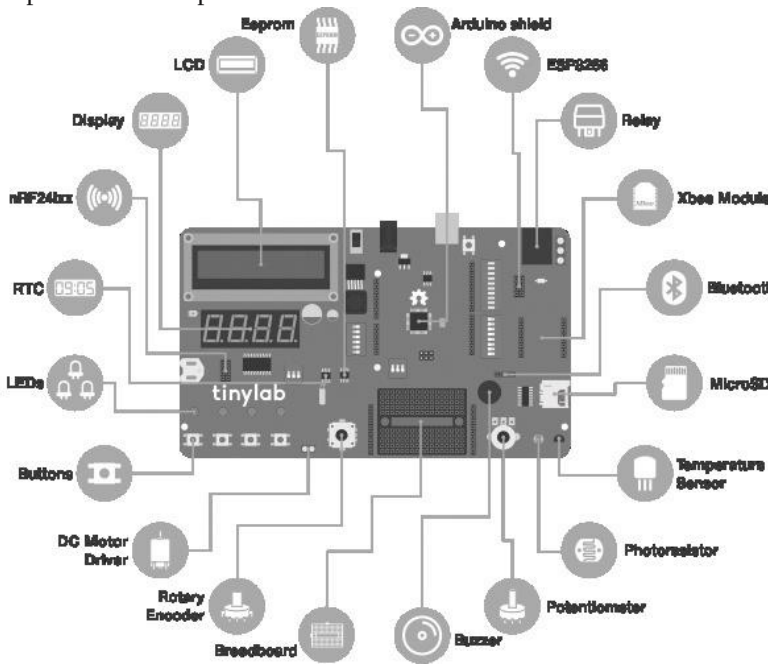
All Arduino boards, as mentioned, offer more than just the basic functionality of blinking an LED on their own. You will need additional components such as circuit elements, motors, sensors, communication modules, etc., to fully utilize your Arduino board.

Arduino starter kits are available for this purpose, typically containing the necessary materials and a breadboard for assembling circuits using these materials. Additionally, if you plan to use your Arduino for more project-focused purposes, there are "shields" available that come with pre-built circuits ready to be attached onto the Arduino board. During the learning process, circuits are often assembled on a breadboard, connections are made with the Arduino, and quite frequently, the circuit doesn't work on the first attempt due to a connection error. This is a natural part of the learning process and is something that happens to everyone.

Especially for beginners learning Arduino, it can be challenging to determine whether an error is in the software or in the hardware and connections. Tinylab addresses this issue by providing a single board with commonly used Arduino modules or a socketed structure that can be easily mounted. This practical usage method helps prevent connection problems that may occur during circuit assembly.

Tinylab not only benefits beginners but also provides great convenience for those who have been using Arduino for a long time as a practical and portable prototyping platform. You'll find it very useful for testing your projects, especially IoT applications, as it neatly houses all the necessary

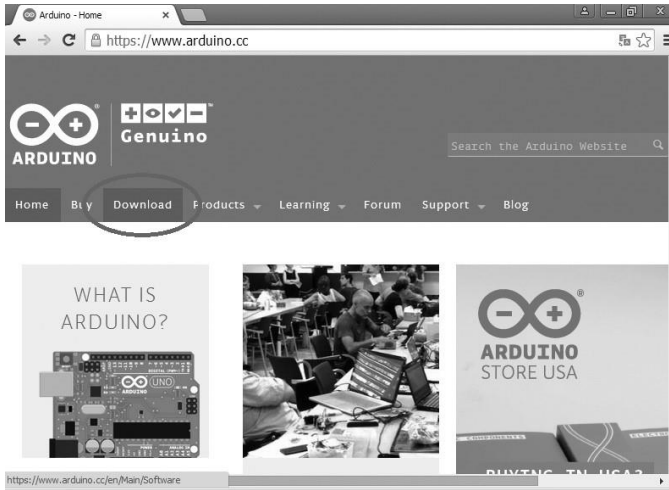
components in one place.



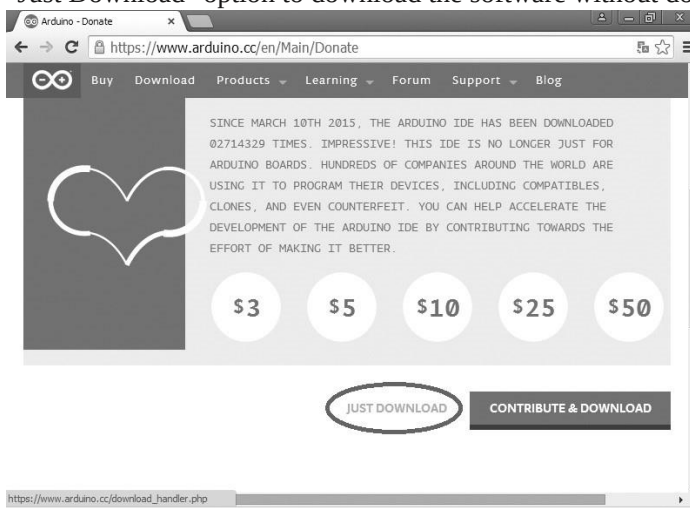
Installation of Arduino IDE and Drivers

To be able to use our Tinylab, we need a programming environment and the necessary drivers installed on our computer. My recommendation is to install the Arduino software and the necessary drivers on your computer before connecting your Tinylab to avoid any driver-related issues. This way, when you first connect the board to your computer, the installed drivers will work properly.

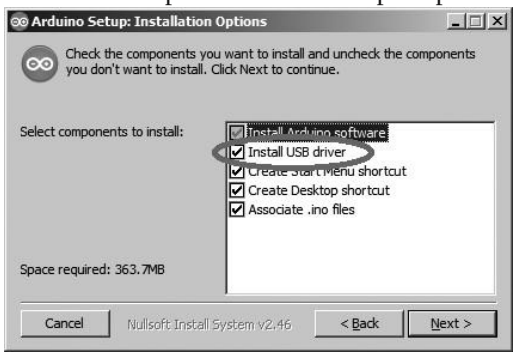
To download the Arduino software, go to www.arduino.cc and navigate to the "Downloads" section.



After clicking on the "Downloads" tab, you will see a screen where you can download the file according to your operating system. At the time of writing this book, the latest version of the Arduino software was 1.6.9. For Windows users, you can click on the "Windows Installer" option. Then, a page requesting donations will open. Depending on your preference, you can choose to make a donation or simply click on the "Just Download" option to download the software without donating.



After that, the software installation file starts to download. Once the download process is complete, we open the file and initiate the installation process. During the installation, we ensure that the "Install USB driver" option is selected if prompted.



Now we can open our Arduino program. After opening the program, the first thing we need to do is to configure the program to work with Tinylab. We click on the Tools > Board menu and select Arduino Leonardo.

Next, we select the port where our Arduino is connected from the Tools menu > Port submenu. Note that the port number may vary on each computer.

Now we have an Arduino program ready for use. In the program, the functions written in the void setup() section run only once when the board receives power. We set the input/output pins to be used, serial port configuration, and other settings in this section. The void loop() section contains the functions that will run continuously until the board loses power after the commands in the setup function have been executed.

After writing our program, when we want to upload it to our board, we first click on the "Verify" option. The program asks us to save the code to a directory on our computer first, and then it compiles the code and notifies us of any errors. If there are no errors in our code and our

Arduino board is connected to our computer via USB, we can click on the "Upload" option to upload our code to the board.

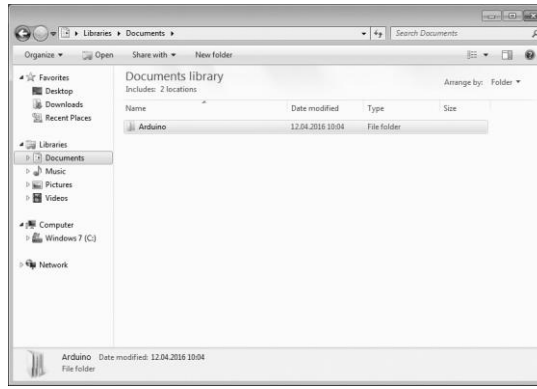
Libraries:

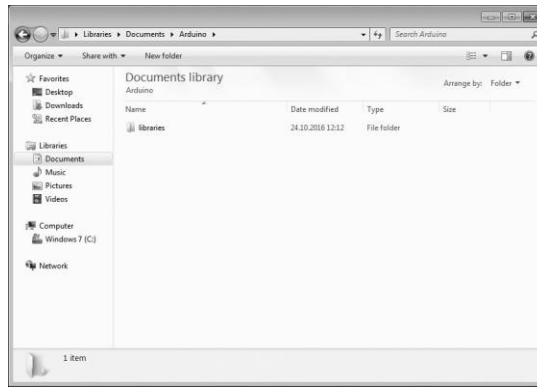
Arduino's own features and external hardware components, modules, shield boards, etc., have ready-made codes called "libraries". Libraries take over most of the heavy lifting from our code, such as communication protocols, pin definitions, function definitions, etc., required for our external hardware to work. We can easily use that external hardware by simply including the relevant library in our code (for example, an LCD screen).

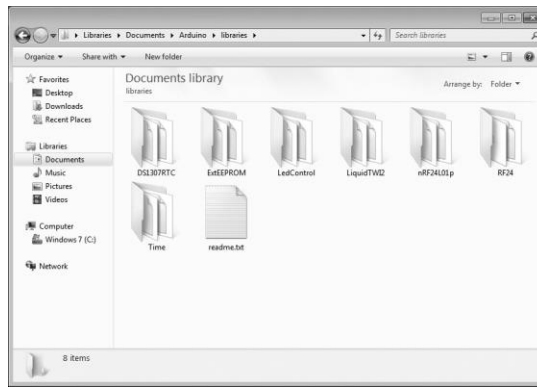
For using components such as the LCD screen, real-time clock (RTC), 7-segment display controller, external EEPROM, etc., found on Tinylab, we need separate library files. The necessary libraries for the applications included in the book are provided at the following link: http://maker.robotistan.com/download/tinylab_libraries.zip

To install the libraries, copy the "libraries" directory downloaded from the above link to the Arduino directory on your computer (for Windows and Mac OS, it's usually Documents -> Arduino):

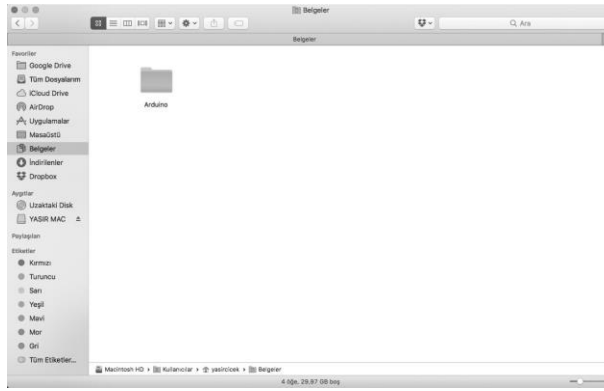
For Windows:


Libraries ▸ Documents ▸
Search Documents
Organize
Open
Share with
New folder
Documents library
Includes: 2 locations
Arrange by: Folder
Name
Date modified
Type
Size
Arduino
12.04.2016 10:04
File folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Computer
Windows 7 (C:)
Network
Arduino Date modified: 12.04.2016 10:04
File folder

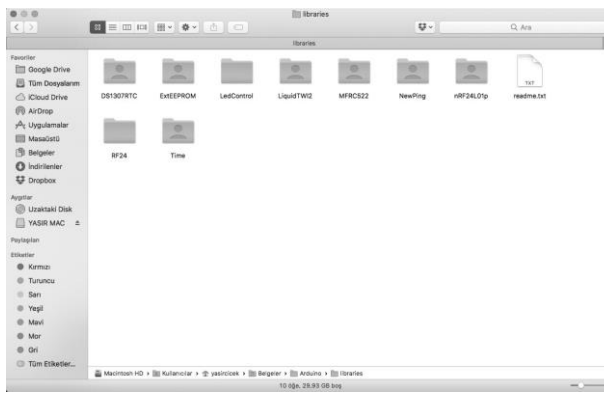

Libraries ▸ Documents ▸ Arduino ▸
Search Arduino
Organize
Share with
New folder
Documents library
Arduino
Arrange by: Folder
Name
Date modified
Type
Size
libraries
24.10.2016 12:12
File folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Computer
Windows 7 (C:)
Network
1 item


Libraries ▸ Documents ▸ Arduino ▸ libraries ▸
Search libraries
Organize
Share with
New folder
Documents library
libraries
Arrange by: Folder
DS1307RTC
ExtEEPROM
LedControl
LiquidTWI2
nRF24L01p
RF24
Time
readme.txt
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Computer
Windows 7 (C:)
Network
8 items

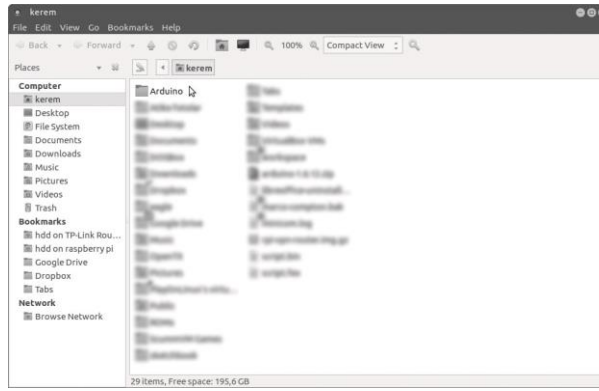
For Mac OS:

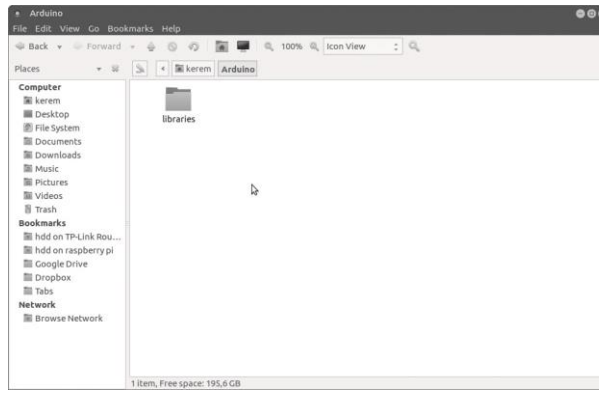

Belgeler
Ara
Belgeler
Favoriler
Google Drive
Tüm Dosyalarım
iCloud Drive
AirDrop
Uygulamalar
Masaüstü
Belgeler
İndirilenler
Dropbox
Aygıtlar
Uzaktaki Disk
YASIR MAC
Paylaşılan
Etiketler
Kırmızı
Turuncu
Sarı
Yeşil
Mavi
Mor
Gri
Tüm Etiketler...
Arduino
Macintosh HD › Kullanıcılar › yasircicek › Belgeler
4 öğe, 29,87 GB boş

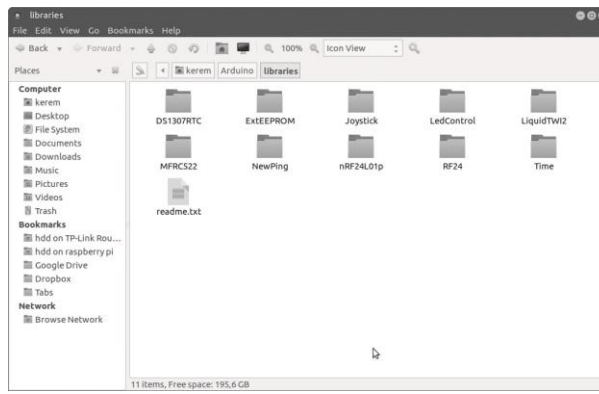

Arduino
Ara
Arduino
Favoriler
Google Drive
Tüm Dosyalarım
iCloud Drive
AirDrop
Uygulamalar
Masaüstü
Belgeler
İndirilenler
Dropbox
Aygıtlar
Uzaktaki Disk
YASIR MAC
Paylaşılan
Etiketler
Kırmızı
Turuncu
Sarı
Yeşil
Mavi
Mor
Gri
Tüm Etiketler...
libraries
Macintosh HD › Kullanıcılar › yasircicek › Belgeler › Arduino › libraries
1 / 1 seçili, 29,96 GB boş


libraries
Ara
libraries
Favoriler
Google Drive
Tüm Dosyalarım
iCloud Drive
AirDrop
Uygulamalar
Masaüstü
Belgeler
İndirilenler
Dropbox
Aygıtlar
Uzaktaki Disk
YASIR MAC
Paylaşılan
Etiketler
Kırmızı
Turuncu
Sarı
Yeşil
Mavi
Mor
Gri
Tüm Etiketler...
DS1307RTC
ExtEEPROM
LedControl
LiquidTWI2
MFRC522
NewPing
nRF24L01p
readme.txt
RF24
Time
Macintosh HD › Kullanıcılar › yasircicek › Belgeler › Arduino › libraries
10 öğe, 29,93 GB boş

For GNU/Linux (Ubuntu MATE 16.04 LTS is used):


kerem
File Edit View Go Bookmarks Help
Back Forward 100% Compact View
Places kerem
Computer
kerem
Desktop
File System
Documents
Downloads
Music
Pictures
Videos
Trash
Bookmarks
hdd on TP-Link Rou...
hdd on raspberry pi
Google Drive
Dropbox
Tabs
Network
Browse Network
Arduino
29 items, Free space: 195,6 GB


Arduino
File Edit View Go Bookmarks Help
Back Forward 100% Icon View
Places kerem Arduino
Computer
kerem
Desktop
File System
Documents
Downloads
Music
Pictures
Videos
Trash
Bookmarks
hdd on TP-Link Rou...
hdd on raspberry pi
Google Drive
Dropbox
Tabs
Network
Browse Network
libraries
1 item, Free space: 195,6 GB


libraries
File Edit View Go Bookmarks Help
Back Forward 100% Icon View
Places kerem Arduino libraries
Computer
kerem
Desktop
File System
Documents
Downloads
Music
Pictures
Videos
Trash
Bookmarks
hdd on TP-Link Rou...
hdd on raspberry pi
Google Drive
Dropbox
Tabs
Network
Browse Network
DS1307RTC
ExtEEPROM
Joystick
LedControl
LiquidTWI2
MFRC522
NewPing
nRF24L01p
RF24
Time
readme.txt
11 items, Free space: 195,6 GB

2

Lessons

If we have successfully completed the setup stage, we can now begin our lessons.

Our first lesson will be the simplest example, which is to turn on and off an LED.

Lesson 01: Blinking LED

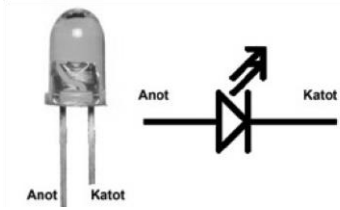
For those who do not have Tinylab, here are the necessary materials:

- Breadboard
- Arduino
- 1 x LED
- 1 x 220Ω Resistor
- Jumper Cables

What is LED?

LED stands for Light Emitting Diode, which means a light-emitting diode. Unlike the small bulbs we are familiar with and use in most of our projects, LEDs have two different legs called anode and cathode. The

anode should be connected to positive voltage, represented by the + symbol, while the cathode should be connected to negative voltage, represented by the - symbol or ground (GND).



LED ve devre sembolü

Voltage, Current and Ohm's Law

We know that various electrical devices operate at different voltages. Our Arduino board operates at 5V voltage. However, the situation is a bit different for our LED. It is essential that the maximum current passing through the LED does not exceed 20 mA (milliamperes = one-thousandth of an ampere). We mentioned that our Arduino operates at 5V. The 5V value represents the output voltage of the board. However, the LED requires a current of 20 mA. It seems things are starting to get a little confusing. No need to worry! There is a solution to everything.

If we were to connect our LED directly to Arduino, the LED would have the maximum amount of current provided by the board flowing through it, potentially damaging either our LED or our board. To prevent this, we need to connect a resistor in series with our LED to reduce the current. Now, how do we determine the value of this resistor? This is where Ohm's Law comes into play:

$$\mathbf{V = i \times R}$$

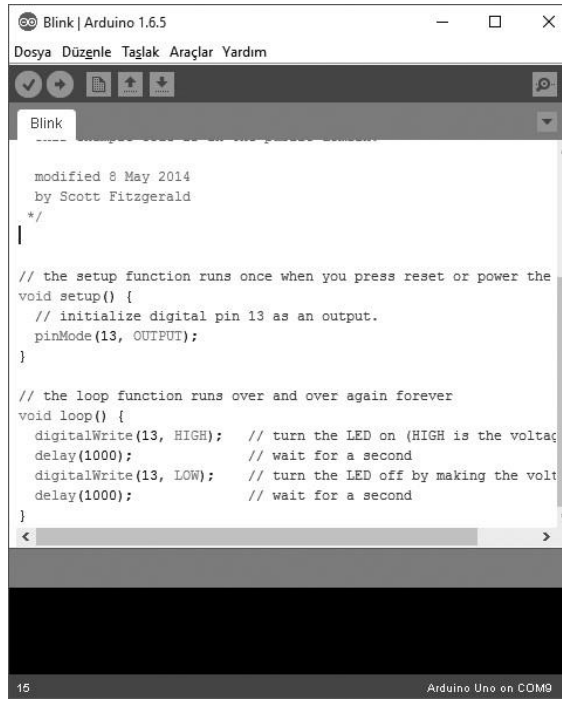
In this equation, V represents voltage, I represents current, and R represents resistance. If we were to connect the LED, which requires 20 mA of current, to one of the pins of our Arduino board supplying 5V:

We would obtain the equation:

$$\mathbf{5V = 0{,}020A \times R}$$

If we solve for R in this equation, we get the result of 250. This means that we need a resistor with a value of 250 Ω (ohms) to use our LED with a 5V voltage. It's not crucial to get the exact value; we can use the 220 Ω resistor that we have available. That's enough theoretical information for now. Now let's prepare our Arduino program. There are 4 LEDs connected to digital output pins 10, 11, 12, and 13 on our Tinylab board. We don't need any external connections or resistors to use these LEDs. Additionally, most Arduino boards have an LED connected to pin 13 with a series resistor next to it. This LED is typically marked with the letter "L" on the board. Let's open our Arduino IDE program and follow these steps to open the "Blink" example program:

File -> Examples -> 01.Basics > Blink



Let’s review this code togerther:

```
pinMode(13, OUTPUT);
```

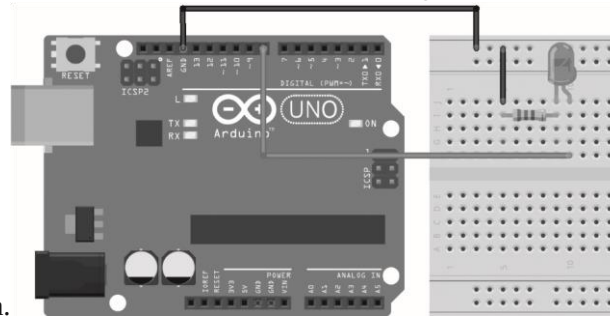
This line sets pin 13 on the board as an output. If we don't specify whether the pin will be used as an input or output, any input or output functions we write later in the program won't be able to use that pin.

```
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);
```

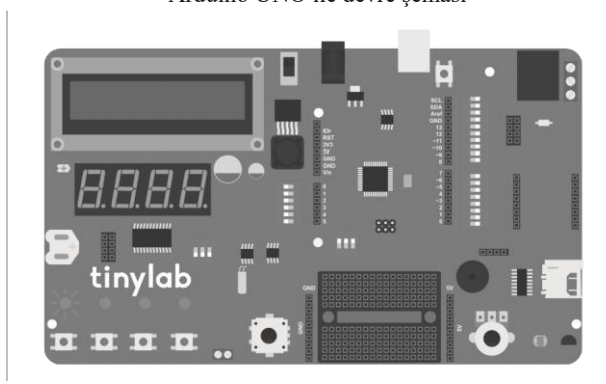
This part first sets pin 13 to the HIGH logic level, which is 5V, waits for 1000 milliseconds (1 second) without performing any operations, and

then sets pin 13 to the LOW logic level, which is 0V or ground level. After completing this process, the microcontroller waits again for 1 second without performing any operations using the delay function.

By changing the durations of the delay commands in this code, we can alter the amount of time the LED remains on and off. If we want to use a different pin, all we need to do is replace the pin number in the pinMode and digitalWrite functions with the desired pin number. In the Tinylab board, we can use LEDs connected to pins 12, 11, and 10 in addition to pin 13. If we are making an external connection, we should remember to include a 220 Ω resistor in series when connecting the LED to the Arduino pin.



Arduino UNO ile devre şeması



Tinylab kartında “Blink” programı çalışırken

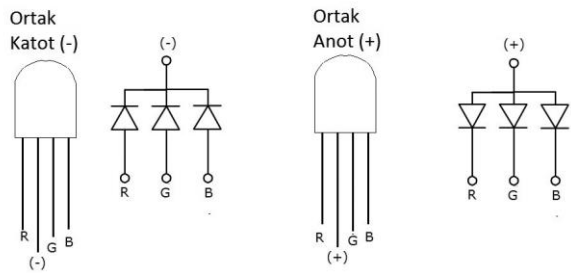
Lesson 02: RGB LED Control

For those who do not have a Tinylab, the necessary materials are:

- Arduino
- Breadboard
- 1 x Common Anode or Common Cathode RGB LED
- 3 x 220Ω Resistor
- Jumper Cable

If you are going to use Tinylab for the lesson, an RGB LED, 3 pieces of 220Ω resistors, and jumper cables will be sufficient. You can build the circuit on the breadboard located on the board.

RGB LEDs, unlike regular LEDs, contain 3 different colored LEDs (red, green, and blue) in a single package. As you may recall from our previous lesson, LEDs have anode and cathode terminals. In RGB LEDs, the anode or cathode connections are common, depending on the manufacturing method of the LED.



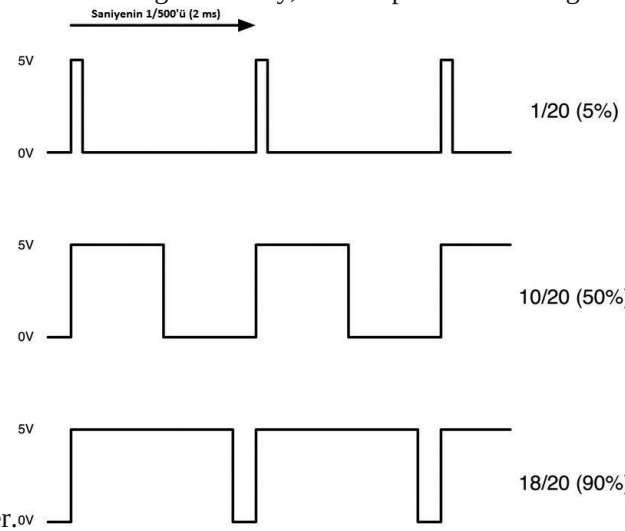
Our RGB LED in the kit we're using has a common anode. Therefore, the code we will prepare for Arduino will work according to the common anode configuration. If we want to use a common cathode LED, we will need to make a slight modification to our code.

PWM and Digital to Analog Conversion

As we know, the voltage used on the input/output pins of our Arduino is at a 5V level. In our previous lesson, we connected our LED to our board in a way that it drew 20 mA of current at 5V voltage, resulting in the LED being as bright as possible. But what if we want to change the

brightness? The solution is simple: We need to decrease the voltage. If we operate our LED, which works with 5V, with a lower voltage, such as 3V, the brightness will decrease. However, you may ask: Wasn't the Arduino output voltage 5V? How can we get a 3V output?

Here is where pulse width modulation (PWM) comes into play. Instead of using the term "pulse width modulation," I'll use the abbreviation PWM. PWM allows us to obtain any voltage between 0 and 5V from the output pin of the Arduino by turning the 5V voltage on and off within a certain time interval (usually 1/500th of a second on Arduino boards). Think of it this way: If an LED flashes on and off very quickly, our eyes cannot perceive this flashing accurately, and we perceive the brightness as being lower.

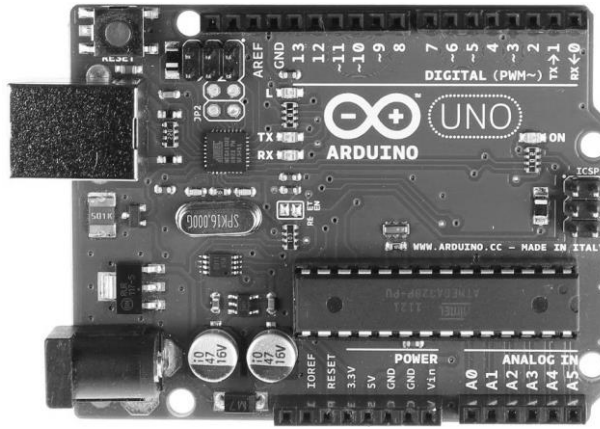


%5, %50 ve %90 duty cycle değerlerine sahip PWM sinyalleri

As seen in the image above, if we provide 5V but only during 2 milliseconds, which is just 5% of the total time, the resulting value will be 5% of 5V, which is 0.25V. Similarly, if we set it to be open for half of the 2ms period (50%), we get 2.5V. The percentage of time the signal is open within one period is called the duty cycle.

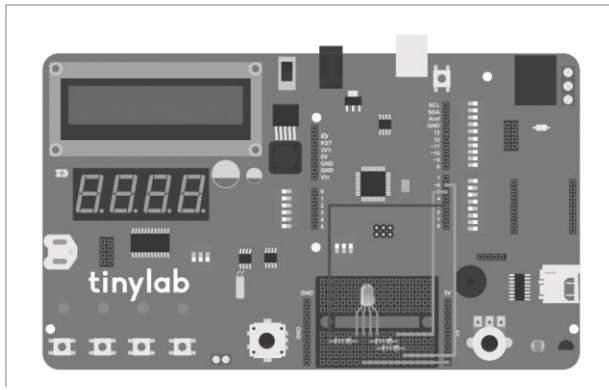
Not all pins of our Tinylab board have PWM output capability. Some pins on our board have a ~ sign in front of the pin number. We must use these

pins if we want PWM output. For Tinylab, which is Arduino Leonardo, these pins are 3, 5, 6, 9, 10, 11, and 13.



Arduino UNO kartındaki PWM çıkış sunabilen dijital pinler

Since our Tinylab board does not have a built-in RGB LED, we make our connections on the mini breadboard according to the following diagram:

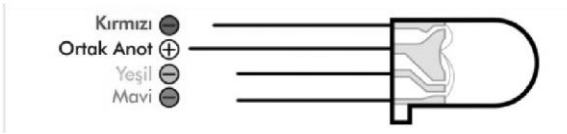


In our first lesson, we modified the existing sample code within the Arduino software. This time, we are writing our own code:

```
//pin tanimlamalari
 #define kirmiziPin 3
```

```
#define yesilPin 5
#define maviPin 6

void setup() {   //pinleri
cikis olarak ayarla
pinMode(kirmiziPin, OUTPUT);
pinMode(yesilPin, OUTPUT);
pinMode(maviPin, OUTPUT); }

void loop() {   renkAyarla(255, 0,
0); //kirmizi   delay(1500);
renkAyarla(0, 255, 0); //yesil
delay(1500);   renkAyarla(0, 0,
255); //mavi   delay(1500);
  renkAyarla(255, 255, 0); //sari
delay(1500);   renkAyarla(80, 0, 80);
//mor   delay(1500);   renkAyarla(0,
255, 255); //acik mavi   delay(1500);
renkAyarla(255, 255, 255); //beyaz
delay(1500); }

void renkAyarla(int kirmizi, int yesil, int mavi) {
//ortak anot icin degerleri tersine cevirmemiz gerekli
kirmizi = 255 - kirmizi;   yesil = 255 - yesil;   mavi =
255 - mavi;   //analog cikislara parlaklik degerlerini
yaz   analogWrite(kirmiziPin, kirmizi);
analogWrite(yesilPin, yesil);   analogWrite(maviPin,
mavi); }
```

We connect the red leg of our RGB LED to pin 3, the green leg to pin 5, and the blue leg to pin 6, each with a 220 Ω resistor. Since the LED is of common anode type, we also connect the anode leg to the 5V pin of our board.



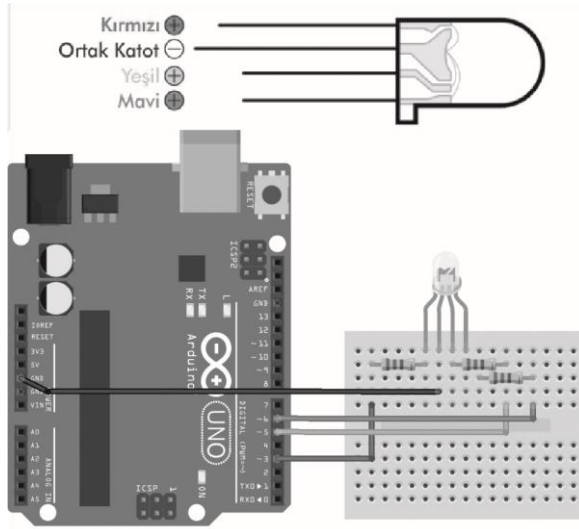
Ortak anot tipindeki RGB LED bağlantı pinleri

In the setup function of our code, we define the pins we will use as outputs. Additionally, the analogWrite command in the setColor function allows us to adjust the voltage level we will receive from each PWM output pin. The analogWrite command is used as follows:

```
analogWrite(PWM çıkış pin numarası, 0-255 arası sayısal değer);
```

In the **`analogWrite`** command, the value 255 represents the maximum output voltage, which corresponds to 5V. All values between 0 and 255 correspond to voltage values between 0 and 5V. For example, the analogWrite(9, 80) command allows us to obtain an output voltage of 5V × (80/255) = 1.57V from pin 9. By mixing different brightness levels of red, blue, and green light, we can achieve the desired color of light. If the LED we are using has a common cathode instead of a common anode, we need to connect the common leg to the GND pins instead of +5V, and we should remove the following lines from the setColor function:

```
  kirmizi = 255 - kirmizi;   yesil = 255 - yesil;   mavi = 255 -
mavi;
```



Ortak katot tipindeki RGB LED bağlantı pinleri

In the loop function, our board repeatedly calls the setColor function we created, allowing us to write the desired values to the outputs. As we know, the delay function causes our board to wait without performing any action between commands. By changing the value of this function, we can achieve transitions between colors at the desired speed. You can also experiment with different brightness values to obtain various colors.

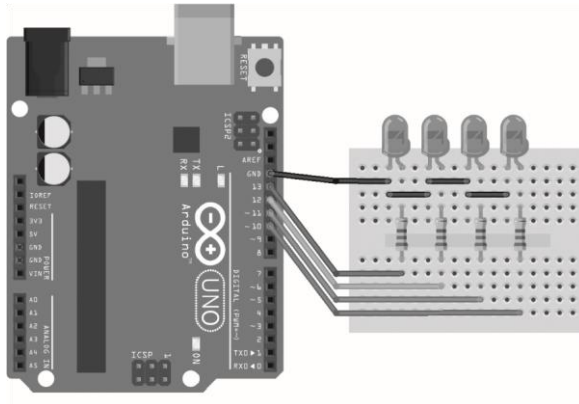
Lesson 03: Serial Communication

Materials needed for those who do not have Tinylab

- Arduino
- Breadboard
- 4 x LED
- 4 x 220 Ω Resistor
- Jumper Cable

If you remember, when our Arduino was connected to the computer, it was recognized as a tty device (COM port for Windows). This tty/COM port is used for serial communication (UART). To communicate between our computer and Arduino, we will use the "Serial Monitor" available in the Arduino software.

Connection Diagram:



The code:

```
//LED pinleri
 #define led1 13
 #define led2 12
 #define led3 11
 #define led4 10

 void setup() {   //LED pinlerini cikis
 olarak ayarla   pinMode(led1, OUTPUT);
 pinMode(led2, OUTPUT);   pinMode(led3,
 OUTPUT);   pinMode(led4, OUTPUT);
 //seri iletisimi 9600 baud'da baslat
   Serial.begin(9600);   //seri monitor acilana kadar bekle
 while (! Serial);   Serial.print("1 ile 4 arasinda bir LED
 numarasi girin veya
");
   Serial.println("x ile hepsini sondurun");
 }

void loop() {    if (Serial.available()) //seri monitor acilana
kadar bekle   {     char ch = Serial.read(); //seri porttan gelen
karakteri oku     switch (ch) //karaktere gore LED'leri yak     {
case '1':
```

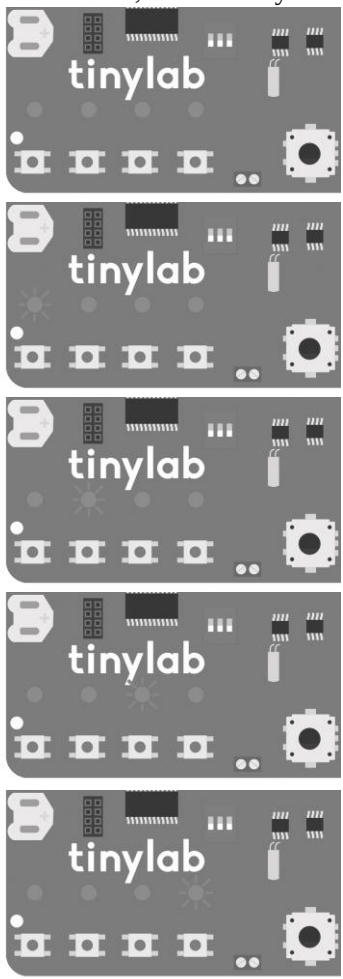
```
        digitalWrite(led1, HIGH);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
digitalWrite(led4, LOW);
Serial.println("1 numarali LED yandi");
break;       case '2':
        digitalWrite(led1, LOW);
digitalWrite(led2, HIGH);
digitalWrite(led3, LOW);
digitalWrite(led4, LOW);
Serial.println("2 numarali LED yandi");
break;       case '3':
        digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, HIGH);
digitalWrite(led4, LOW);
Serial.println("3 numarali LED yandi");
break;       case '4':
        digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
digitalWrite(led4, HIGH);
Serial.println("4 numarali LED yandi");
break;       case 'x':
        digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
digitalWrite(led4, LOW);
Serial.println("Tum LED'ler sondu");
break;
      default: //hatali karakter girisinde uyari yap
Serial.println("Yanlis giris yapildi.");
break;     }
  }
}
```

In the beginning part of our code, we set pins 10, 11, 12, and 13 on the Tinylab as outputs. This allows us to use the 4 LEDs located on the board. The line Serial.begin(9600) in the setup section sets the serial port of our Arduino to operate at 9600 baud (bits/s). The line while (! Serial) prevents the code from proceeding until the serial port communication of our board starts. Thus, when we open the serial port screen via the Arduino IDE, we are greeted with the message "Enter an LED number between 1 and 4 or turn off all with x".

To write any data to the serial port via Arduino, we use the Serial.print or Serial.println functions. Serial.println, unlike the Serial.print command, ensures that a line break is made after the sent text.

To read the data sent to the serial port by the computer, we use the Serial.read function. In this example, we are only reading a single character of data because the number of the LED we want to light up is only one character long. Similarly, the letter "x" we use to turn off all LEDs also occupies only one character. If we enter multiple characters in the serial port screen, for example, typing 123, our Arduino will only detect the last character written (in the example 123, since the last

character is 3, we will only see LED number 3 lit up).



Tinylab üzerindeki L1, L2, L3 ve L4 LED'leri

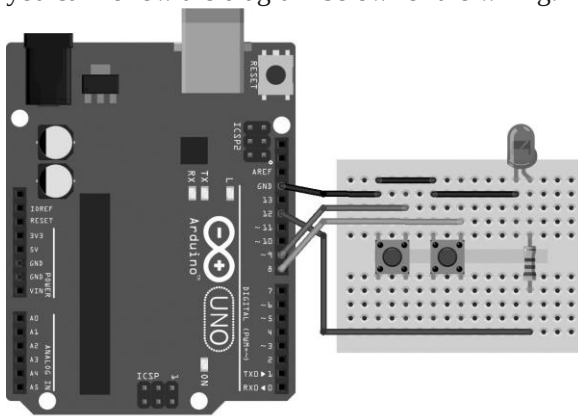
Lesson 04: Digital Inputs

Materials needed for those who do not have Tinylab

- Arduino
- Breadboard
- 1 x LED

- 1 x 220 Ω resistor
- 2 x push buton
- Jumper cable

Arduino board has both digital and analog pins. Until now, we have used digital pins only for output functions. However, digital pins can also be used with input devices such as sensors and buttons. In this lesson, we will learn how to turn an LED on and off using two push buttons. Tinylab has 4 buttons onboard. Buttons S1 and S2 are connected to digital pins 9 and 8, respectively. If you are making external connections, you can follow the diagram below for the wiring:



The code:

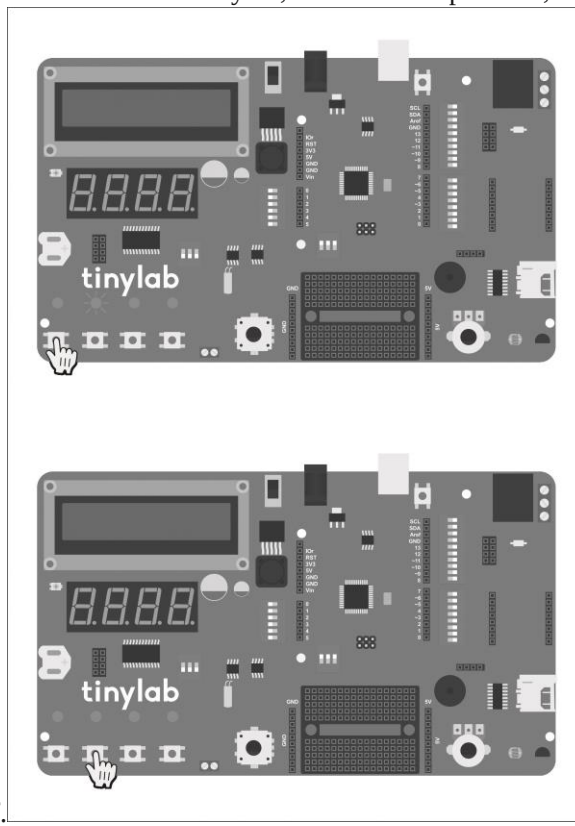
```
#define ledPin 12
#define buttonApin 9
#define buttonBpin 8

void setup() {   //LED i cikis, butonlari
giris olarak ayarla   pinMode(ledPin, OUTPUT);
pinMode(buttonApin, INPUT_PULLUP);
pinMode(buttonBpin, INPUT_PULLUP);  }

void loop() {   //A butonuna
basilinca LED i yak   if
```

```
 (digitalRead(buttonApin) == LOW)   {
 digitalWrite(ledPin, HIGH);   }   //B
 butonuna basilinca LED i sondur   if
 (digitalRead(buttonBpin) == LOW)   {
 digitalWrite(ledPin, LOW);   }
}
```

If you noticed, when defining the pins where the buttons are connected, we used a declaration like INPUT_PULLUP instead of just INPUT. By doing this, we activate the pull-up resistor integrated into the digital pins of the Arduino board. But what does the pull-up resistor do?

The pull-up resistor ensures that the signal remains intact when we use digital pins as inputs. In this project, when the button is not pressed, the value read from the digital pin is 5V, which corresponds to the logic HIGH level. The pull-up resistor ensures that the voltage at this pin remains at 5V as long as the button is not pressed and the value is not pulled LOW. In this lesson, when we press button S1 on our Tinylab, the

L2 LED will turn on and stay on, and when we press S2, this LED will



turn off.

Lesson 05: Analog Inputs

Materials needed for those who do not have Tinylab

- Arduino
- Breadboard
- 1 x LM35 temperature sensor
- 1 x 10kΩ potentiometer
- 1 x 5mm LDR
- 1 x 10kΩ resistor
- Jumper cable

So far, we've mainly used the digital input/output pins of our Arduino. However, you may have noticed that Arduino boards also have an "Analog Input" section. We can use these pins to read voltage by converting analog to digital.

Analog-to-Digital Converter, ADC

The microcontroller on our Arduino board has an analog-to-digital converter (ADC) with 10-bit resolution. But what does this 10-bit resolution mean? As we know, our Arduino's microcontroller operates at 5V. The 10-bit ADC in this microcontroller can read voltages between 0V and 5V with a precision of 210, which equals 1024 steps. This means that a voltage of 0V applied to one of the analog input pins corresponds to a digital value of 0, while a voltage of 5V corresponds to a digital value of 1023.

Although it's possible to use voltage levels lower than 5V with the analog-to-digital converter in the microcontroller, we won't need this capability since all analog inputs on the Tinylab board (potentiometer, light sensor, and temperature sensor) operate at a 5V level.

Potentiometer

The potentiometer is actually a component found in almost all devices we use daily. For example, the knob we turn to adjust the volume of our music system is typically connected to a potentiometer. In simple terms, a potentiometer is a resistor that we adjust by turning it with our hands. In microcontroller applications, it is commonly used as a voltage divider.

When we turn the potentiometer in one direction, the resistance value between its two terminals changes. We can observe this by measuring it with a multimeter.

The potentiometer on our Tinylab board is connected to the A0 analog

input.

Light Dependent Resistor

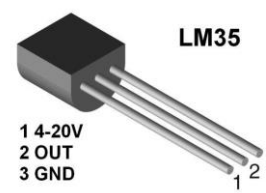
The photoresistor, also known as a light-dependent resistor (LDR), is a component with resistance that varies depending on the intensity of light it receives. Its resistance decreases as the amount of light falling on it increases. It is commonly used in many electronic devices we use daily, often referred to as a "photocell."

The LDR on our Tinylab board is connected to the A2 analog input.

Temperature Sensor

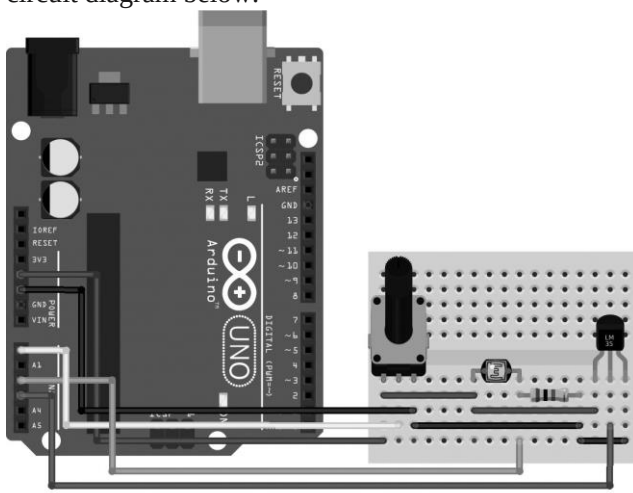
The LM35 temperature sensor is an analog output temperature sensor that provides precise temperature measurements. It has a sensitivity of 0.5°C at 25°C. This sensor can be connected to Arduino's analog input to perform temperature measurement applications.

The temperature sensor on our Tinylab is connected to the A3 analog input.



All analog sensors on Tinylab, as mentioned earlier, will output a numerical value between 0 and 1023. However, we will need some conversion formulas to be able to measure parameters such as temperature, light level, etc., using these values.

If you don't have a Tinylab, you can set up your circuit according to the circuit diagram below:



The code:

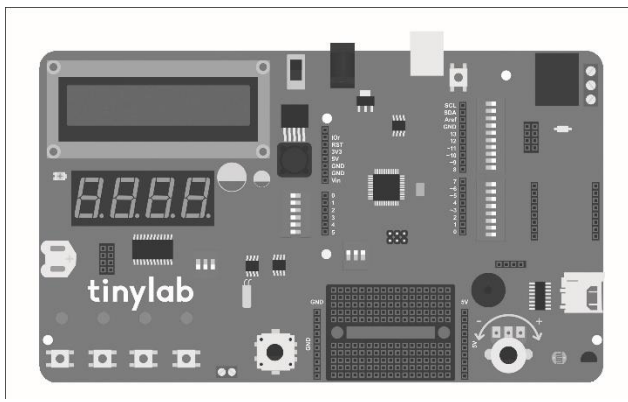
```
//sensor pinleri
#define pot_pin A0
#define ldr_pin A2
#define lm35_pin A3

void setup() {
  //seri iletisimi 9600 baud'da baslat
  Serial.begin(9600);   //seri monitor acilana
kadar bekle   while (! Serial);
Serial.println("Analog giris okuma ornegi.");
delay(500); }
```

```
void loop() {   //potansiyometreyi oku   int pot_val =
analogRead(pot_pin);   //ldr yi oku ve degeri lux e cevir
int ldr_val = ((2500.0 / (analogRead(ldr_pin) * (5.0 /
1024.0))) - 500) / 10.0;    //lm35 i oku ve degeri
santigrada cevir
  int lm35_val = (5.0 * analogRead(lm35_pin) * 100.0) / 1024;
Serial.print("Potansiyometre degeri: ");
  Serial.println(pot_val);
  Serial.print("LDR degeri: ");
  Serial.println(ldr_val);
  Serial.print("Sicaklik degeri: ");
Serial.println(lm35_val);   delay(100);
}
```

The output we receive from pin 2 of our LM35 temperature sensor corresponds to a voltage drop of 10mV per 1°C. We need to scale the 5V analog input range according to this ratio. We accomplish this with the following line of code in our program:

`int lm35_val = (5.0 * analogRead(lm35_pin) * 100.0) / 1024;`

Similarly, to convert the voltage from the LDR into a meaningful lux value representing the amount of ambient light, we use the following line of code: **`int ldr_val = ((2500.0 / (analogRead(ldr_pin) * (5.0 / 1024.0))) - 500) / 10.0;`** For the data coming from the potentiometer, we directly measure it as a value between 0 and 1023.



Tinylab üzerindeki potansiyometre, breadboard'un hemen sağında yer alır.

Lesson 06: Button Controlled RGB LED

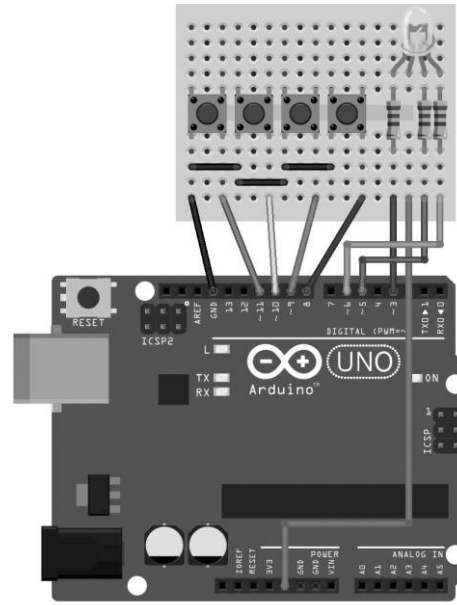
Materials needed for those who do not have Tinylab

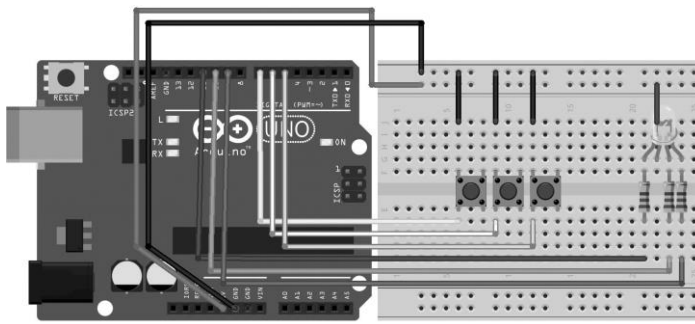
- Arduino
- Breadboard
- 1 x common anode or common cathode RGB LED
- 3 x 220Ω resistor
- 4 x push-button
- Jumper cable

Previously, we've worked on an RGB LED project together, and we've also learned how to control LEDs using push buttons in our digital input lesson. We can think of this project as a combination of those two previous ones. We will use three buttons to control the brightness of the RGB LED for its red, green, and blue colors, and another button to turn off the LED altogether. This way, we can achieve different colors each time without having to change the code.

The connection of the RGB LED is the same as the schema we used in Lesson 2.

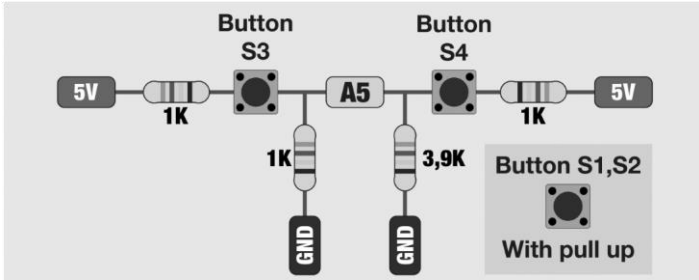
If you don't have a Tinylab, you can connect the buttons according to the following diagram:





Here, there is an important point to note: The S3 and S4 buttons on the Tinylab are connected to analog pin A5 with resistors instead of

digital pins.



Tinylab üzerindeki S3 ve S4 butonlarının bağlantı şeması.

As a result, using a single analog pin, a total of 3 button connections are provided (the encoder button is also connected using this method). The values in the provided code are read with a margin of error of ±20 (which corresponds to approximately a 2% margin of error in a 10-bit range). If these values are causing issues in your application, you can first read the values from pin A5 and adjust them according to the values suitable for your board.

If you're using your own Arduino board instead of Tinylab, there is no such limitation, so you can connect the button inputs to pins 8, 9, 10, and 11. However, in this case, you'll need to make changes to the code accordingly.

The code for Tinylab:

```
//LED baglantilari
#define kirmiziLEDPin 3
#define yesilLEDPin 5
#define maviLEDPin 6
//buton baglantilari
#define kirmiziSwitchPin 9
#define yesilSwitchPin 8
#define maviSwitchPin A5
```

```
//varsayilan degerler
int kirmizi = 0; int
mavi = 0; int yesil =
0;

void setup() {   //ledleri cikis, butonlari
giris olarak ayarla   pinMode(kirmiziLEDPin,
OUTPUT);   pinMode(yesilLEDPin, OUTPUT);
pinMode(maviLEDPin, OUTPUT);
pinMode(kirmiziSwitchPin, INPUT_PULLUP);
pinMode(yesilSwitchPin, INPUT_PULLUP); }

void loop() {   //kirmizi butonuna basildigi surece
degeri birer arttir   if (digitalRead(kirmiziSwitchPin)
== LOW)   {     kirmizi ++;
    if (kirmizi > 255) {
kirmizi = 0;     }
  }   //yesil butonuna basildigi surece degeri birer arttir
if (digitalRead(yesilSwitchPin) == LOW)   {     yesil ++;
if (yesil > 255) {       yesil = 0;     }
  }   //mavi butonuna basildigi surece degeri birer arttir
if (analogRead(maviSwitchPin) >= 185 &&
analogRead(maviSwitchPin) <= 225)   {     mavi ++;     if
(mavi > 255) {       mavi = 0;     }
  }   //S4 e basildiginda sondur   if
(analogRead(maviSwitchPin) >= 489 &&
analogRead(maviSwitchPin) <= 529)   {     kirmizi
= 0;     yesil = 0;     mavi = 0;   }
renkAyarla(kirmizi, yesil, mavi);   delay(10); }

void renkAyarla(int kirmizi, int yesil, int mavi)
{
  //ortak anot icin degerleri tersine cevirmemiz gerekli
kirmizi = 255 - kirmizi;   yesil = 255 - yesil;   mavi = 255 -
mavi;   //analog cikislara parlaklik degerlerini yaz
```

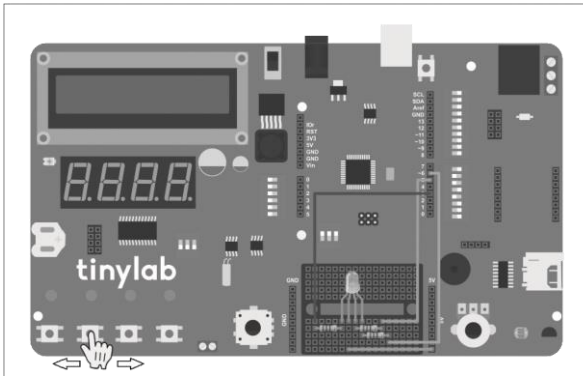
```
analogWrite(kirmiziLEDPin, kirmizi);   analogWrite(yesilLEDPin,
yesil);   analogWrite(maviLEDPin, mavi); }
```

The code for your own Arduino:

```
//LED baglantilari
#define kirmiziLEDPin 3
#define yesilLEDPin 5
#define maviLEDPin 6
//buton baglantilari
#define kirmiziSwitchPin 11
#define yesilSwitchPin 9
#define maviSwitchPin 10
#define resetSwitchPin 8

//varsayilan degerler int
kirmizi = 0; int mavi = 0;
int yesil = 0;

void setup() {   //ledleri cikis, butonlari giris
olarak ayarla   pinMode(kirmiziLEDPin, OUTPUT);
pinMode(yesilLEDPin, OUTPUT);   pinMode(maviLEDPin,
OUTPUT);   pinMode(kirmiziSwitchPin, INPUT_PULLUP);
  pinMode(yesilSwitchPin, INPUT_PULLUP);
}
void loop() {   //kirmizi butonuna basildigi surece degeri
birer arttir   if (digitalRead(kirmiziSwitchPin) == LOW)   {
kirmizi ++;     if (kirmizi > 255) {       kirmizi = 0;     }
  }   //yesil butonuna basildigi surece degeri birer arttir
if (digitalRead(yesilSwitchPin) == LOW)   {     yesil ++;
if (yesil > 255) {       yesil = 0;     }
  }   //mavi butonuna basildigi surece degeri birer arttir
if (digitalRead(maviSwitchPin) == LOW)   {     mavi ++;
if (mavi > 255) {       mavi = 0;     }
  }   //sondurmeye basildiginda sondur   if
(digitalRead(resetSwitchPin) == LOW)   {
kirmizi = 0;     yesil = 0;     mavi = 0;   }
renkAyarla(kirmizi, yesil, mavi);   delay(10);
```

```
}

void renkAyarla(int kirmizi, int yesil, int mavi) {
//ortak anot icin degerleri tersine cevirmemiz gerekli
kirmizi = 255 - kirmizi;   yesil = 255 - yesil;   mavi =
255 - mavi;   //analog cikislara parlaklik degerlerini
yaz   analogWrite(kirmiziLEDPin, kirmizi);
analogWrite(yesilLEDPin, yesil);
analogWrite(maviLEDPin, mavi); }
```

If we recall our RGB LED code, we can see that we reused the setColor function in this application. In our project, there is a push button for each color that increases the brightness of that color. As long as these buttons are pressed, the brightness value of the corresponding color increases, and when it reaches 255, it resets back to 0.



S1 ile kırmızı, S2 ile yeşil, S3 ile mavi renklerin parlaklıklarını arttırabilir; S4 ile LED'in tamamen sönmesini sağlayabilirsiniz.

If the LED you are using has a common cathode structure, you can apply the same changes here as we did in our initial RGB project.

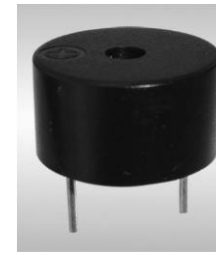
Lesson 07: Sound Output with Buzzer

Materials needed for those who do not have Tinylab

- Arduino
- Breadboard

- 100 Ω resistor
- Buzzer
- Jumper cable

We can think of the component called a buzzer as a small speaker. Although they do not produce sound as loud and detailed as speakers, they are sufficient for generating "beep" sounds.

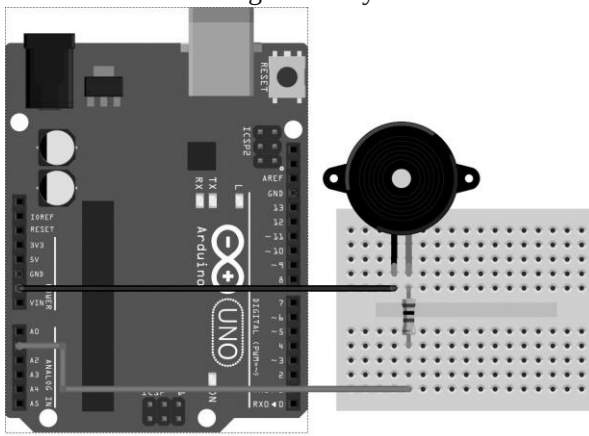


Notes and Frequency

As we know, almost every sound has its own note. Each note also has a specific frequency value. As the frequency increases, the sound becomes higher-pitched. You can find the frequency values for different notes in the following list: Note Frequencies

In this project, we will create a code that plays the do, re, mi, fa, sol, la, and si notes that we remember from elementary school music class. You may recall that notes have two different representations: letter and name. C = do, D = re, E = mi, F = fa, G = sol, A = la, and B = si. Since words like "do" are used by different commands in programming languages, we will use the letter representation of the notes in this code.

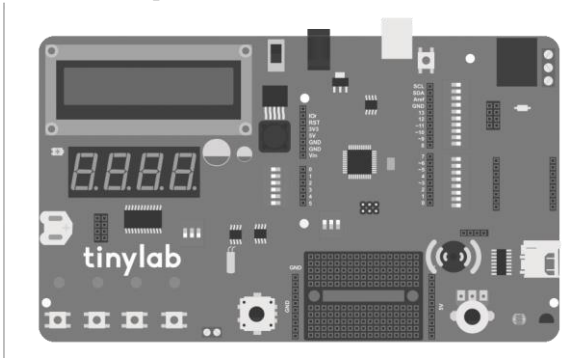
Here is our circuit diagram that you can use with Arduino:



The code:

```
#define buzzerPin A1


int notaSayisi = 8;

//do, re, mi, fa, sol, la, si ve ince do icin frekans degerleri
int C = 262; int D = 294; int E = 330; int F = 349; int G =
392; int A = 440; int B = 494; int C_ = 523;

//notalarin tamamini array icinde tut int
notalar[] = {C, D, E, F, G, A, B, C_};

void setup() {   //0.5sn araliklar ile
sirayla notalari cal   for (int i = 0; i <
notaSayisi; i++)
  {
    tone(buzzerPin, notalar[i]);
delay(500);
noTone(buzzerPin);
delay(20);   }
noTone(buzzerPin); }

void loop()
{

}
```

As you've noticed, our entire code is within the setup function. This means that when we supply power to our Arduino board, the sound will play only once and then stop. If we want the sound to play again, all we need to do is press the reset button on the Arduino.



Tinylab üzerindeki buzzer, breadboard'un sağ üst köşesinde, reset butonu ile USB bağlantısının hemen sağında yer alır.

To explain our code, we first input the frequency value for each note. Then, we arranged these notes from the low C note to the high C note into an array. We used a for loop to iterate through the contents of this array and used the tone command for sound output. Once the tone command is given, Arduino continues to output sound until it encounters the noTone command.

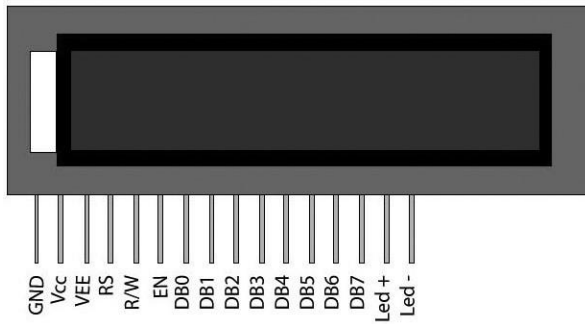
Lesson 08: 16x2 LCD Screen

Materials needed for those who do not have Tinylab

- Arduino
- Breadboard
- 16×2 LCD Screen
- 1 x 10 kΩ potentiometer
- 1 x 5mm LDR
- 1 x LM35 temperature sensor
- Jumper cable

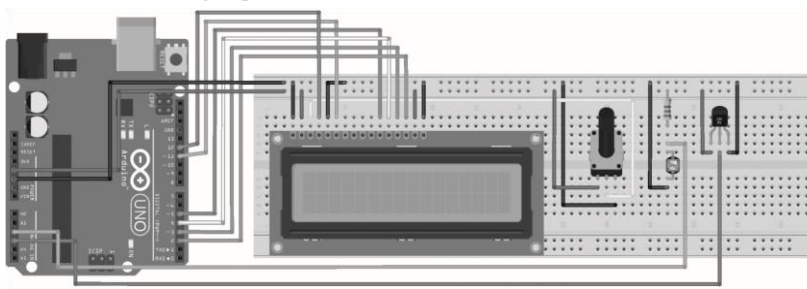
In this project, we will display text on an LCD screen. These LCD screens are commonly seen in vending machines, home alarms, and

many other places. They typically consist of 2 lines with 16 characters each and are quite popular in the microcontroller world. However, when we directly connect these LCDs to our microcontroller, they occupy at least 6 pins (register select, enable, and 4 data connection pins).



Alt açıklama: 16x2 dizilimine sahip LCD ekranın pin bağlantıları.

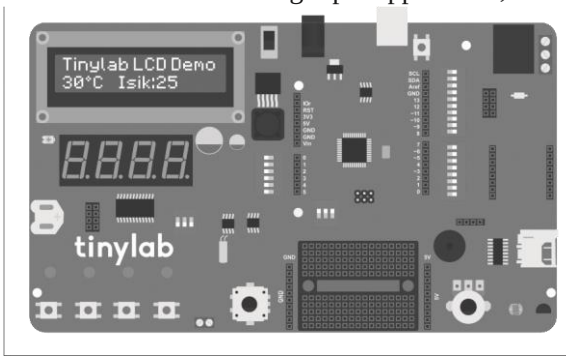
If you're connecting a standard 16-pin LCD screen with your own Arduino, you can set up your circuit according to the following wiring diagram. For the connection of the LDR and temperature sensor, you can refer to our analog input lesson.



If you want to check your screen connections, you can access the sample code by following the steps File -> Examples -> LiquidCrystal -> HelloWorld.

On Tinylab, to avoid restricting the input/output pins, an LCD screen is used with the MCP23008 input/output expander integrated circuit that operates with the I2C protocol. This allows all LCD functions to be fully utilized with only 2 connection pins. The only thing we need to pay attention to is including the library required for using this driver in the

code. Additionally, we should include the Wire.h library in our code, which enables I2C communication with Arduino. The library required for the LCD driver is included in the package mentioned in the installation instructions in the early parts of our book. We can access the sample applications that come with the library under File -> Examples -> LiquidTWI2 (for example, HelloWorld).
In our application, we will print the sensor measurement data, which we will recall from our analog input application, on our LCD.



The code for Tinylab:

```
//gerekli kutuphaneler: I2C icin Wire.h,
//I2C LCD icin LiquidTWI2.h
#include <Wire.h>
#include <LiquidTWI2.h>

//sensor pinleri
#define lm35_pin A3
#define ldr_pin A2

//I2C LCD icin tanimlama LiquidTWI2
lcd(0);

void setup() {   //I2C LCD icin ayar
lcd.setMCPType(LTI_TYPE_MCP23008);
//16 sutun, 2 satir   lcd.begin(16,
2);   //arka isigi ac
lcd.setBacklight(HIGH);   //ilk satira
```

```
Tinylab LCD Demo yaz
lcd.print("Tinylab LCD Demo");
delay(1000); }

void loop() {   //ldr yi oku ve degeri lux e cevir   int
ldr_val = ((2500.0 / (analogRead(ldr_pin) * (5.0 / 1024.0))) -
500) / 10.0;   //lm35 i oku ve degeri santigrada cevir   int
lm35_val = (5.0 * analogRead(lm35_pin) * 100.0) / 1024;
  //imleci ikinci satir birinci karaktere getir
lcd.setCursor(0, 1);   //sicaklik degeri   lcd.print(lm35_val);
//derece sembolunu   lcd.print("\337C");   //imleci ikinci
satir altinci karaktere getir   lcd.setCursor(5, 1);
lcd.print("Isik:");   //isik degeri   lcd.print(ldr_val);
lcd.print(" ");   delay(250); }
```

Our code for the circuit we set up with Arduino:

```
//gerekli kutuphaneler: LCD icin LiquidCrystal.h
#include <LiquidCrystal.h>

//sensor pinleri
#define lm35_pin A3
#define ldr_pin A2

//LCD ekran pinleri sirasiyla RS, E, D4, D5, D6, D7
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {   //16 sutun, 2 satir
lcd.begin(16, 2);   //ilk satira LCD
Demo yaz   lcd.print("LCD Demo");
delay(1000);
}
void loop() {   //ldr yi oku ve degeri lux e cevir   int
ldr_val = ((2500.0 / (analogRead(ldr_pin) * (5.0 / 1024.0)))
- 500) / 10.0;   //lm35 i oku ve degeri santigrada cevir
int lm35_val = (5.0 * analogRead(lm35_pin) * 100.0) / 1024;
//imleci ikinci satir birinci karaktere getir
```

```
lcd.setCursor(0, 1);   //sicaklik degeri
lcd.print(lm35_val);   //derece sembolunu
lcd.print("\337C");   //imleci ikinci satir altinci karaktere
getir   lcd.setCursor(5, 1);   lcd.print("Isik:");   //isik
degeri   lcd.print(ldr_val);   lcd.print(" ");   delay(250);
}
```

The parts located in the setup section of the code are making the necessary configurations for our LCD driver library. Additionally, the backlight of our LCD can also be turned on and off through a command (`lcd.setBacklight`).

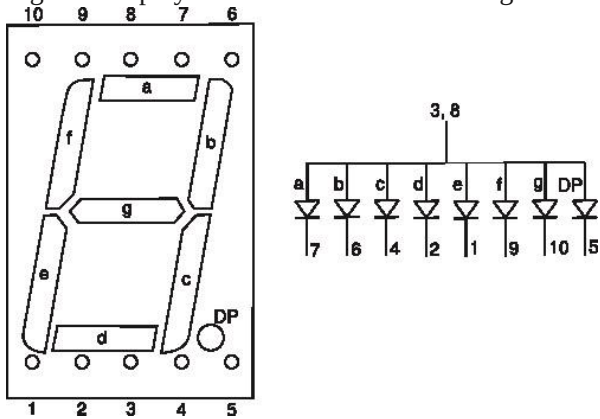
Lesson 09: 7 Segment LED Display

Materials needed for those who do not have Tinylab

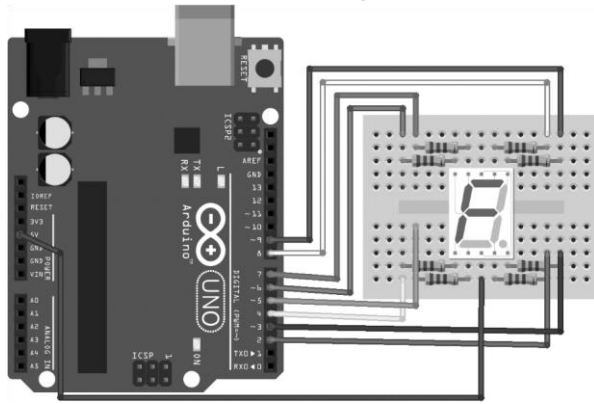
- Arduino
- Breadboard
- 8 x 220Ω resistor
- 1 x 7 segment display
- Jumper cable

A 7-segment display is a type of numerical display commonly found in elevator floor indicators, digital clocks, kitchen appliances, and numerous other applications. It can display numeric characters as well as some alphanumeric characters. The operating principle is quite simple: to form a digit, the display consists of 7 (sometimes 8, including the decimal point) LEDs. By lighting up specific LEDs, we can create the desired digit. Similar to RGB LEDs, there are two types: common anode and common cathode. Below, you can see the arrangement of a 7-

segment display in a common cathode configuration.

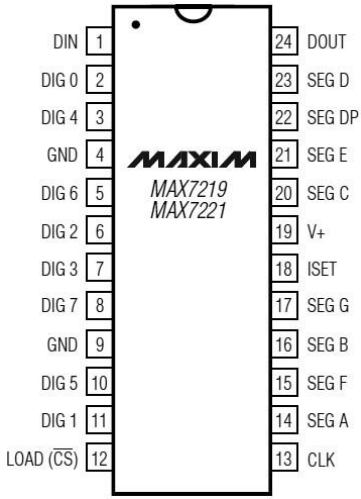


- As we can see from the diagram above, a single-digit 7-segment display requires 8 LED connections. Working with single-digit displays using a logic similar to that of RGB LEDs is straightforward:



However, as you can see, it occupies quite a lot of pins. Assuming we have 13 digital outputs on our Arduino board, it seems impossible to directly connect even 2 digits. But there's a very simple solution to this: using driver ICs. To drive the 4-digit 7-segment display found in Tinylab, the MAX7219 integrated circuit is used. This allows all 4 digits

of the display to be controlled using just 3 digital pins.



MAX7219 entegresinin pinleri.

To use our MAX7219 integrated circuit in our Arduino code, we need to add the library file named LedControl.h to our code. Additionally, since the MAX7219 integrated circuit on our Tinylab board is connected to digital pin 10 for DataIn, digital pin 12 for Clock, and digital pin 11 for LOAD, our configuration line is written as follows:

```
LedControl lc=LedControl(10, 12, 11, 1);
```

The number 1 at the end of the configuration indicates that we will control only 1 7-segment display. After ensuring that the switches located just below the 7-segment display on our Tinylab are in the up position, we can upload our code.

The code:

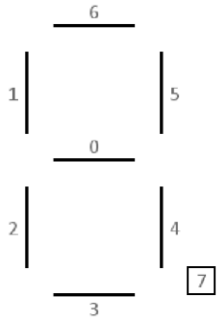
```
//gerekli kutuphaneler: MAX7219 7-segment surucu icin
//LedControl.h
#include <LedControl.h>
//sicaklik sensoru lm35 A3'e bagli
#define lm35_pin A3
```

```
//DataIn, Clock, Load, surucu sayisi
LedControl lc = LedControl(10, 12, 11, 1);

//baslangic degerleri int
temp = 0; int
temp_first_digit = 0; int
temp_second_digit = 0;

void setup() {   //gostergeyi calistir
lc.shutdown(0, false);   //gosterge
parlakligini 0-15 arasinda ayarla
lc.setIntensity(0, 8);   //gostergeyi temizle
  lc.clearDisplay(0); }

void loop() {   //lm35 i oku ve degeri santigrada cevir   temp
= (5.0 * analogRead(lm35_pin) * 100.0) / 1024;   //gostergeye
“tiny” yaz   lc.setRow(0, 0, B00001111);   lc.setRow(0, 1,
B00000110);   lc.setRow(0, 2, B01110110);   lc.setRow(0, 3,
B00111011);   delay(1000);   //bir sonraki islemden once
gostergeyi temizle   lc.clearDisplay(0);   //gostergeye “lab”
yaz   lc.setRow(0, 0, B00001110);   lc.setRow(0, 1, B01111101);
lc.setRow(0, 2, B00011111);   delay(1000);   //bir sonraki
islemden once gostergeyi temizle   lc.clearDisplay(0);
//gostergeye sicaklik degerini yaz   if (temp <= 10) //sicaklik
tek basamakli ise   {     temp_first_digit = temp; //birler
basamagina sicakligi     temp_second_digit = 0; //onlar
basamagina 0 yaz   }   //birler basamagini hesapla
temp_first_digit = temp / 10;   //onlar basamagini hesapla
temp_second_digit = temp % 10;   //birler basamagini gostergeye
yaz   lc.setDigit(0, 0, temp_first_digit, false);
  //onlar basamagini gostergeye yaz
  lc.setDigit(0, 1, temp_second_digit,
false);   //gostergeye santigrad derece
sembolunu yaz   lc.setRow(0, 2, B01100011);
lc.setRow(0, 3, B01001110);   delay(1000); }
```

In our setup function, we give the command lc.shutdown(0, false) for the operation of our display, and then we set the screen brightness with the command lc.setIntensity(0, 8). The lc.clearDisplay(0) command is used to clear any residual text on the display.

You may recall the temperature calculation formula from our analog inputs lesson in the loop function. The next part focuses on displaying the desired numbers or letters on the 7-segment display.

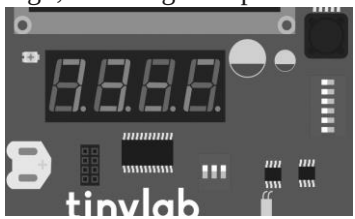
When introducing the 7-segment display, I mentioned that it has a total of 8 LEDs. So how can we control these LEDs? The answer is simple: 1 byte of data corresponds to 8 bits. In the data of 1 byte size for each digit, we can set the desired segments of that digit as on or off. For this, we use the table below:

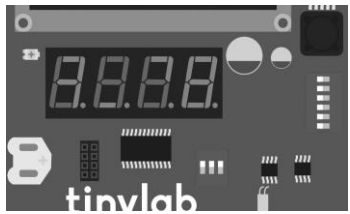


B	0	0	0	0	0	0	0	0
	▼	▼	▼	▼	▼	▼	▼	▼
	7	6	5	4	3	2	1	0

Using this table, we can create the letter "t" very conveniently as follows: B00001111. Similarly, for the letter "I" we use B00000110, for the letter "n" we use B01110110, and finally for the letter "y" we use B00111011. In the lc.setRow(a, b, Bxxxxxxxx) function, the "a" part indicates which 7-segment display we will use, with 0 representing the first one, and the "b" part indicates which digit we will use. Digit 0 represents the leftmost

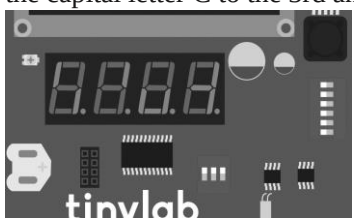
digit, while digit 3 represents the rightmost one.





With the method described above, we can control each LED of each digit individually. However, there is a much easier method for entering numbers: the lc.setDigit(a, b, number, false) function. The usage of this function is very similar to lc.setRow. Similarly, it is used to determine which display and digit will be used for the first two parameters. Then, the number that follows is used to write a digit between 0 and 9 to that digit. Of course, another problem arises here: How do we write each digit of a number one by one?

Fortunately, the solution to this is quite easy as well. For the units digit of the number, we take its remainder when divided by 10 (mod 10), for the tens digit, we take its division by 10, for the hundreds digit, we divide it by 100, and for the thousands digit, we divide it by 1000. In our application, we write the temperature value to the first two digits of our 7-segment display, and then we sequentially write the degree sign and the capital letter C to the 3rd and 4th digits.



The small "if" statement in the code ensures that when the temperature value is less than 10, it is written to the units digit, as numbers are written from left to right.

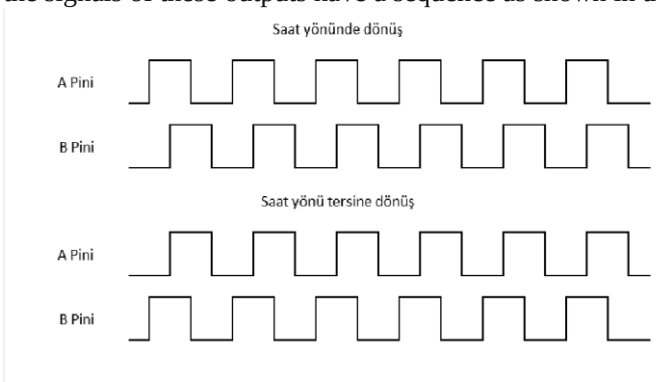
Ders 10: Rotary Encoder

Materials needed for those who do not have Tinylab

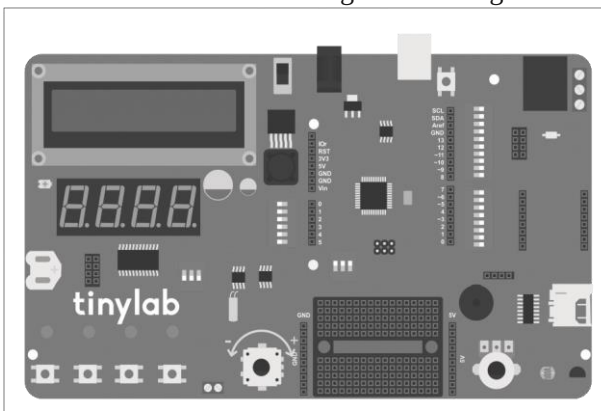
- Arduino
- Breadboard
- Rotary encoder with button
- 10kΩ resistor
- Jumper cable

Rotary encoders can be used in microcontroller applications for incremental control tasks in button and potentiometer forms (for example, LED brightness, sound level, and motor speed control). They can also be used with DC motors to obtain position feedback of the motor shaft. This enables DC motors to be used like servo motors by receiving position feedback.

Rotary encoders have two outputs, A and B. When we turn the encoder, the signals of these outputs have a sequence as shown in the table below:



Simply put, by measuring the signals coming from the A and B pins, we can determine the direction of rotation and the number of revolutions of the encoder based on which signal is leading.



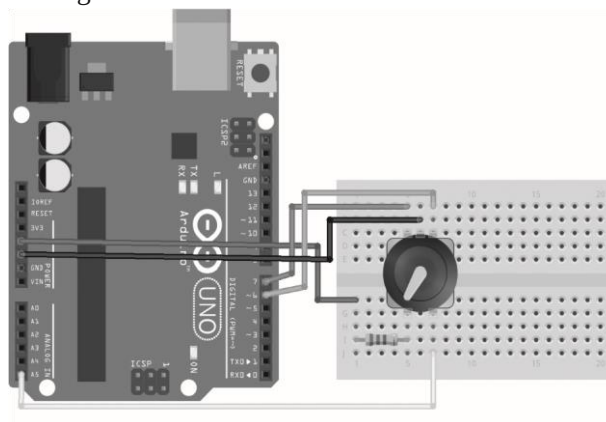
Tinylab üzerinde encoder, breadboard'un sol kısmında yer almaktadır.

The code:

```
//encoder baglanti pinleri
#define pinA 6
#define pinB 7

//baslangic degerleri
int encoderPos = 0;
bool pinALast = LOW;
bool n = LOW;

void setup() {   //enkoder pinlerini giris olarak ayarla,
pull-up direnclerini aktiflestir   pinMode (pinA,
INPUT_PULLUP);   pinMode (pinB, INPUT_PULLUP);   //seri
haberlesmeyi 9600 baud'da baslat
  Serial.begin (9600);
}
void loop() {   //enkoderin A pinini oku   n =
digitalRead(pinA);   //A pini 0 dan 1 e gectiyse   if
```

```
((pinALast == LOW) && (n == HIGH))   {     if
(digitalRead(pinB) == LOW) //B pini 0 daysa
encoderPos++; //enkoder degerini arttir     else
encoderPos--; //eknoder degerini azalt     //enkoder
degerini seri porta aktar
    Serial.print("Enkoder konumu: ");
    Serial.println(encoderPos);   }   pinALast = n; //A pininin
en son durumunu n'e esitle   //S4 e basildiginda enkoder
degerini sifirla   if (analogRead(A5) >= 80 && analogRead(A5)
<= 100)   {     encoderPos = 0;
    Serial.println("Enkoder sifirlandi!");
  }
}
```

Connection Diagram:



On Tinylab, the A pin of the encoder is connected to digital pin 6, and the B pin is connected to pin 7. Therefore, we define the connections at the beginning of the code. We define a variable named encoderPos to store the current position of our encoder, a variable named pinALast to hold the previous position of pin A, and finally, a variable named n to hold the current state of pin A.

In the setup function, we set the A and B pins as INPUT_PULLUP. The reason for using INPUT_PULLUP for inputs is to prevent noise and bouncing that may occur during transitions between LOW and HIGH levels of the encoder. We start serial communication at 9600 baud, and in the loop function, we start our main operation.

In the loop function, first, we read the A pin using the digitalRead function. We keep this value in the n variable and compare it with the previous state of pin A. If the previous value of pin A was LOW and its current value is HIGH, it indicates that the encoder has turned. To determine in which direction the encoder has turned, we need to know the order in which the A and B inputs become HIGH. Therefore, if the B pin is LOW, we increment the encoder position value, and if it is HIGH, we decrement it.
After this process, we write the encoder position value to the serial port screen so that we can see the position of the encoder. To reset the encoder position value, we will use the button on the encoder. If you recall our button-controlled RGB LED lesson, the encoder button was connected to pin A5 with a resistor. The value from pin A5 should be between 80 and 100 (it may vary on each board, you can load the AnalogReadSerial example first and read the value from pin A5 when the encoder button is pressed).

Lesson 11: DC Motor Speed Control

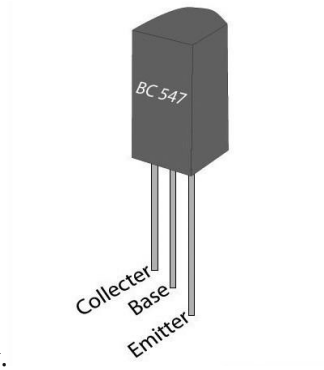
Materials needed for those who do not have Tinylab

- Arduino
- Breadboard
- 220Ω resistor
- BC547 NPN transistor

- 1N4007 diode
- 9V DC motor
- F-F jumper cable
- 9V DC adapter or 9V battery

In this application, we will control the speed of our DC motor using a BJT (Bipolar Junction Transistor) type transistor. But why do we need a transistor?

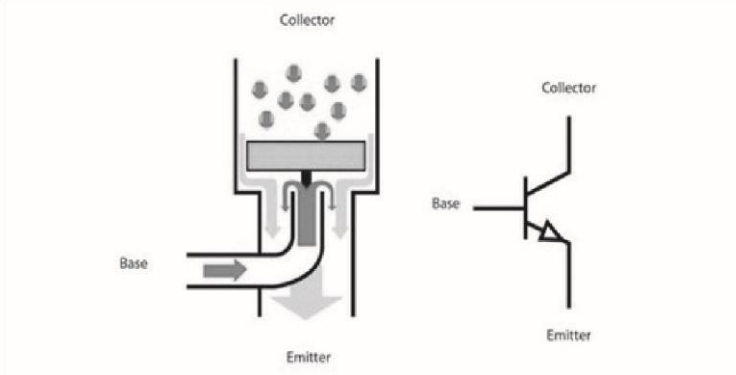
We know that the output voltage we will receive from the digital pins of our Arduino is 5V. Our 9V DC motor can operate with a 5V voltage, but the current it will draw is significantly higher than the maximum current we can draw from Arduino pins, which is 40 mA. This is where the transistor comes into play.



TO-92 kılıfında NPN tipi BC547 transistör.

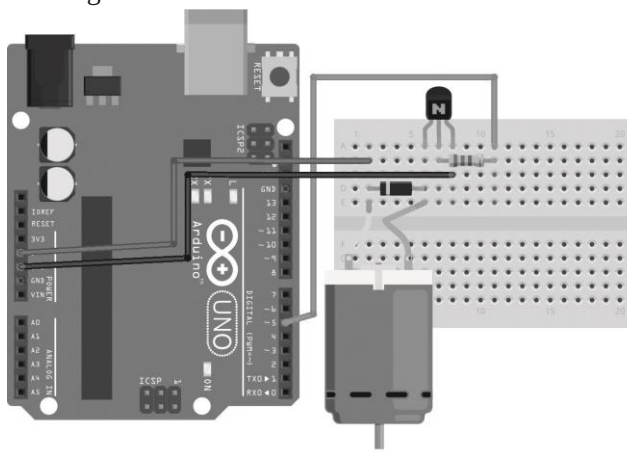
We can think of a transistor as a switch that we can control with electrical current or voltage in a very simple way. To understand this, let's imagine the NPN-type transistor we have as a faucet. Let's imagine that water flows from the collector (collector) leg to the emitter (emitter) leg through a pipe. With the current we give from the base leg, just like controlling the flow of water with a faucet, we can control the amount of

current flowing from the collector to the emitter.



This allows us to control devices that draw large currents using very small currents.

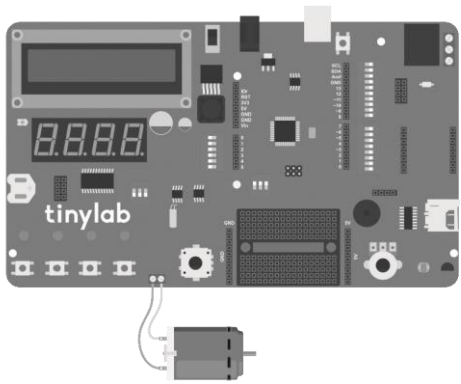
Connection Diagram:



One of the pins of the motor output on our Tinylab board is connected to the GND line via an NPN-type transistor. The other pin of the motor output is connected to the +5V line. The base of the NPN transistor is connected to digital pin 5. By providing a PWM signal to this pin, we can change the current passing through the transistor and easily control the speed of the motor.

Since the motor output is connected to the +5V line, the current from our computer's USB port may be insufficient. Therefore, when using devices that draw high currents like motors, let's make sure to power our Tinylab with an external adapter or battery.

We upload the code to our Tinylab, disconnect the connection, connect the motor to our board, and power our board with an adapter. You can choose not to connect the USB cable to your computer if you wish.



Tinylab'e motoru görselde gösterilen yerdeki vidalı klemensler ile bağlayabilirsiniz.

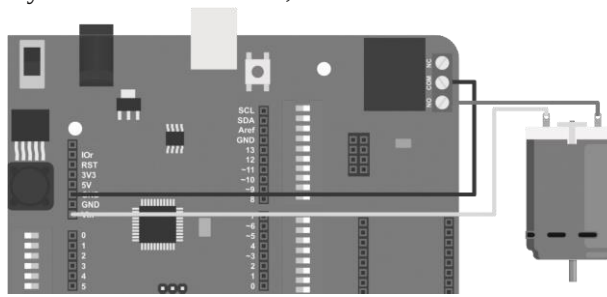
The code:

```
//motor surucu dijital pin 5 e bagli
#define motorPin 5

//baslangic degerleri int hiz = 0;  void setup() {
//motor surucu pinini cikis olarak ayarla
pinMode(motorPin, OUTPUT); }   void loop() {   //0 dan 255
e kadar kademeli olarak motorun hizini arttir   for(hiz =
0; hiz <= 255; hiz++)   {
analogWrite(motorPin,hiz);     delay(20);
}   //255 e ulasinca motoru yavaslatarak durdur   for(hiz =
255; hiz>=0; hiz--)   {
analogWrite(motorPin,hiz);              delay(20);
}
}
```

You may recall the analogWrite function from our RGB LED lessons. We use this function to increase or decrease the voltage we receive from digital pin 5, thereby changing the current flowing into the base of the transistor. As the current flowing into the base of the transistor increases, the current flowing between the collector and emitter increases, thus increasing the speed of our motor.

Thanks to the for loops we use, the speed value gradually increases between 0 and 255 in the analogWrite function, and when it reaches 255, it gradually decreases until it becomes 0. This way, we observe the motor gradually accelerating from a stationary state, reaching maximum speed, and then gradually decelerating in the same incremental manner.

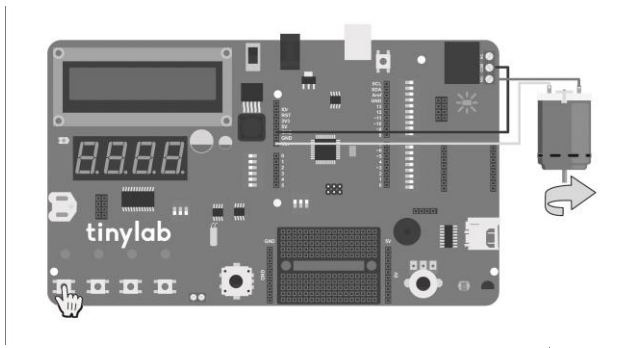
If we want to directly start and stop the motor, we can use the relay located on the Tinylab board. In this case, our connections should be as follows:

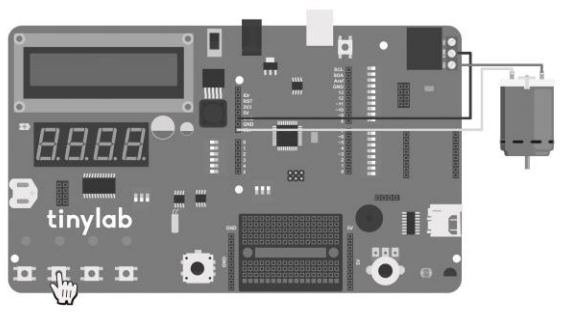


I'd like to talk about the connections labeled NO, NC, and COM on the relay outputs. The terminal labeled COM is the common connection terminal. We connect the connection we want to switch to the COM terminal of our relay, and if we want it to remain closed (open circuit) when there is no power to the relay in our circuit, we make the external circuit connection through the NO (normally open) terminal. If we want the circuit to operate (short circuit) when there is no power to the relay, we should use the NC (normally closed) terminal instead of NO.

The code we will upload is quite simple. We will make the relay energize with the S1 button (D9) on the Tinylab and de-energize it with the S2 button (D8):

```
//buton ve role pinleri
#define relayPin A4
#define buttonApin 9 #define
buttonBpin 8

void setup() {   //role pinini cikis, butonlari giris olarak
ayarla   pinMode(relayPin, OUTPUT);   pinMode(buttonApin,
INPUT_PULLUP);    pinMode(buttonBpin, INPUT_PULLUP);  }

void loop() {   //A butonuna basilinca roleyi iletime
sok   if (digitalRead(buttonApin) == LOW)   {
digitalWrite(relayPin, HIGH);   }   //B butonuna
basilinca roleyi kesime sok   if
(digitalRead(buttonBpin) == LOW)   {
digitalWrite(relayPin, LOW);   }
}
```





Lesson 12: Real Time Clock (RTC)

Materials needed for those who do not have Tinylab

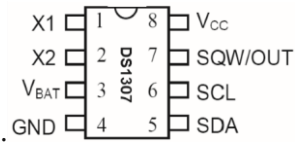
- Arduino
- Breadboard
- 1 adet DS1307 RTC entegre
- 1 adet 0.1μF capacitor
- 2 x 10kΩ resistor
- 1 x 32.768kHz crystall
- 1 x CR2023 battery holder
- 1 x CR2023 battery

Microcontrollers like Arduino obtain the required clock frequency for their operations from circuits that generate pulses, such as crystal oscillators or RC oscillators. Arduino boards typically operate at clock frequencies such as 8 and 16 MHz. The expression "16 MHz" here means that the microcontroller can handle 16 million clock pulses per second. This enables components like timers and registers inside the microcontroller to function correctly.

Functions like delay() and millis() used in Arduino operate based on this clock frequency. However, the accuracy of this frequency may not be sufficient for regular timekeeping. To overcome this problem, real-time clock (RTC) integrated circuits are used. These integrated circuits are also found on the motherboard of personal computers, and the operating system learns the system time from this integrated circuit. If the battery on the motherboard is depleted or removed, you may receive an error indicating that the computer forgot the time and date when booting up. Similarly, the RTC integrated circuit on Tinylab also requires an external battery to remember the time and date when there is no power to the board.

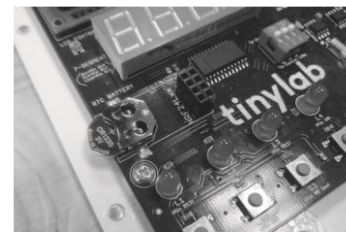
Thanks to the precise crystal oscillators on the RTC integrated circuit, it can accurately measure time (the error margin of an average RTC integrated circuit is about 1 second per year). In this lesson, we will learn

how to create a clock application using the DS1307 integrated circuit on Tinylab.



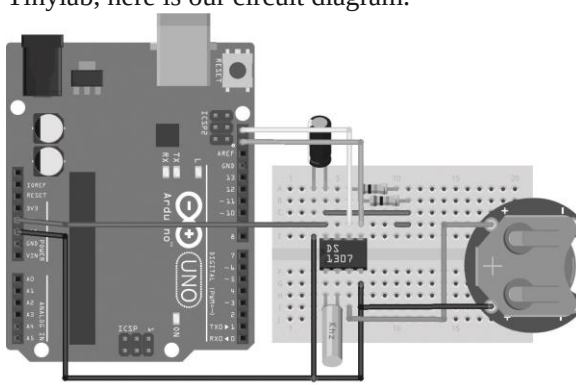
DS1307 entegresinin bağlantı pinleri.

Before starting the application, we place the clock battery that comes with Tinylab into the socket labeled "RTC BATTERY" on the board, with the + side (flat side) facing upwards.



Our next step after placing the battery is to synchronize the RTC's clock with our computer's clock. We will perform this process only once during the initial setup. Once the clock is set correctly, it will continue to operate until the battery runs out.

For those who will build their own circuit with Arduino instead of Tinylab, here is our circuit diagram:



We upload the code found under File -> Examples -> DS1307RTC -> SetTime to our Tinylab. After uploading the code, we open the serial port monitor. If we see a statement like "DS1307 configured Time=16:25:43, Date=Aug 12 2016," it means that the clock of our RTC integrated circuit has been set.

The code:

```
//gerekli kutuphaneler: I2C haberlesme icin Wire.h,
//saat fonksiyonlari icin Time.h,
//DS1307 RTC entegresi icin DS1307RTC.h,
#include <Wire.h>
#include <TimeLib.h> #include
<DS1307RTC.h>

void setup() {
  //seri iletisimi 9600 baud'da baslat
  Serial.begin(9600);   //seri monitor acilana
kadar bekle   while (!Serial);   delay(200);
Serial.println("DS1307RTC Okuma Testi");
  Serial.println("----------------------"); }

void loop() {   //tm formatinda saat bilgisini okumak icin bu
satiri ekliyoruz   tmElements_t tm;

  if   (RTC.read(tm))   //RTC   entegresinden   tm   formatinda
okunabiliyorsa   {
    //saat formati: SS:DD:ss
Serial.print("Saat = ");
print2digits(tm.Hour);
Serial.write(':');
print2digits(tm.Minute);
Serial.write(':');
print2digits(tm.Second);     //tarih
formati: GG/AA/YYYY     Serial.print(",
Tarih (G/A/Y) = ");
   Serial.print(tm.Day);
    Serial.write('/');
    Serial.print(tm.Month);
    Serial.write('/');
```

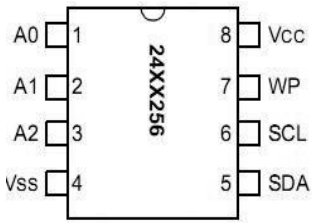
```
    Serial.print(tmYearToCalendar(tm.Year));
    Serial.println();   }   else //hata mesajlari   {     if
(RTC.chipPresent()) //RTC okunuyor fakat saat ayarli degilse
{
      Serial.println("RTC saati ayarli degil.");
Serial.println("Lutfen once SetTime programi ile saati
ayarlayin.");       Serial.println();     }     else //RTC
okunamiyorsa     {
      Serial.println("RTC okuma hatasi.");
     Serial.println();
}     delay(9000);   }
delay(1000); }

void print2digits(int number) //saati iki haneli olarak yaz {
if (number >= 0 && number < 10) //saat tek basamakli ise
onlar basamagina 0 koy
  {
    Serial.write('0');
  }
   Serial.print(number);
}
```

Lesson 13: External EEPROM

EEPROM stands for Electrically Erasable Programmable Read Only Memory, which is a type of integrated circuit used in microcontroller circuits for permanent data storage. It can be likened to the hard drives in computers or USB flash drives. With a microcontroller, we can write desired data to EEPROM and read it whenever necessary. Generally, microcontrollers have three types of memory: program memory, RAM, and EEPROM. The ATmega32u4 microcontroller, at the heart of the Tinylab board, has a built-in EEPROM memory area of 1KB. Data saved to this area will remain permanently unless we intervene, even if the microcontroller loses power. However, EEPROMs have a limited number of write and read cycles. Excessive read and write operations can lead to EEPROM corruption.

In this application, instead of using the internal EEPROM within the ATmega32u4, we will use the 24LC256 integrated circuit available on the Tinylab board. This offers two advantages: Firstly, the capacity of this external EEPROM is 256 KB, providing more storage space

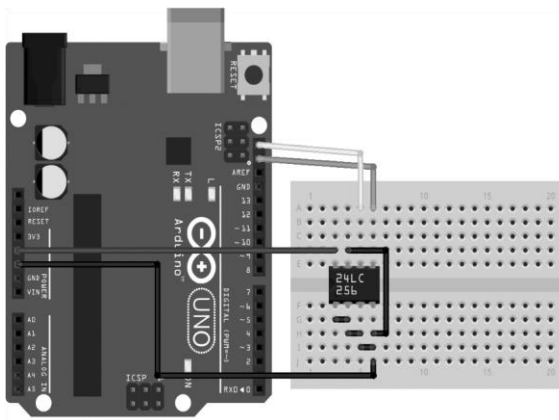
compared to the 1KB EEPROM in the ATmega32u4. Secondly, when the external EEPROM reaches the end of its lifespan, it can be removed and replaced. If the built-in EEPROM in the microcontroller becomes corrupted, the entire microcontroller chip would need to be replaced.



Alt açıklama: 24LC256 entegresinin bağlantı pinleri.

As you can predict from its SDA and SCL pins, the 24LC256 integrated circuit operates with I2C communication. As you may recall from the 16x2 LCD screen lesson, we need to include the Wire.h library in our code for I2C communication. Additionally, for I2C communication, we need to know the address of the device we are communicating with. The 24LC256 integrated circuit can be configured to work at different addresses using the A0, A1, and A2 pins. On the Tinylab board, these pins are connected in a way that allows access to the address 0x50 (all pins are connected to GND).

Below is the schematic for those who will set up their own circuit with Arduino:



We will examine the usage of EEPROM in two parts: writing and reading data. As an example of writing, we will store the information coming from the temperature sensor on the Tinylab into the EEPROM:

```
//gerekli kutuphaneler: 24LC256 icin ExtEEPROM.h, I2C
haberlesme icin Wire.h #include <ExtEEPROM.h>
#include <Wire.h>

//24LC256 I2C adresi
#define hwaddress 0x50
//I2C EEPROM icin tanimlama
ExtEEPROM eeprom(hwaddress);

void setup()
{
  Serial.begin(9600);   Serial.println("EEPROM yazma
ornegi.");   delay(3000);   //adres degeri 256 olana kadar
EEPROM'a sicaklik degerlerini yaz   for (int addr = 0; addr
<= 256; addr++)   {     //lm35 i oku ve degeri santigrada
cevir     int val = (5.0 * analogRead(A3) * 100.0) / 1024;
//degeri EEPROM'un addr ile belirtilen byte'ina yaz
eeprom.write(addr, val);     //suan okunan EEPROM adres
numarasini yaz
    Serial.print(addr);
     //bosluk birak
     Serial.print("\t");
     //EEPROM'a yazilan sicaklik degerini ekrana
yaz     Serial.println(val);     delay(200);   }
  Serial.print("Hafiza sonuna gelindi.");
}


void loop()
{

}
```

I mentioned that the EEPROM integrated on the Tinylab is located at I2C address 0x50. The line #define hwadress 0x50 at the beginning of the code defines the address for I2C communication, thus in this example, we define the address as 0x50. This is a constant address used for communication with the integrated circuit. When discussing what EEPROM is, I mentioned that the 24LC256 has 256KB of memory. This corresponds to a total of 262,144 addresses, each capable of storing data of 1 byte in size (256KB = 256 x 1024 = 262,144 bytes). To write data to each byte of memory, we need to address each memory location individually. A simple for loop inside our loop() function starting from address 0 up to 255 writes a temperature value to one of the EEPROM memory locations every half second.

If you noticed, we are only using the first 256 bytes of the EEPROM. There are 261,888 more addresses available (262,144 - 256) that we could use.

Another point to note is that the data we write must be 1 byte in size. So, if we want to write an integer numerical value, the largest value we can write is 255. This is because a 1-byte data consists of 8 bits. Since 2^8 equals 256, we can only use the range from 0 to 255 for integer-type variables.

To read the data written to the EEPROM, we use the following code:

```
//gerekli kutuphaneler: 24LC256 icin ExtEEPROM.h, I2C
//haberlesme icin Wire.h
#include <ExtEEPROM.h>
#include <Wire.h>

//24LC256 I2C adresi
#define hwaddress 0x50

//I2C EEPROM icin tanimlama
ExtEEPROM eeprom(hwaddress);

void setup()
{
  //seri iletisimi 9600 baud'da baslat
```

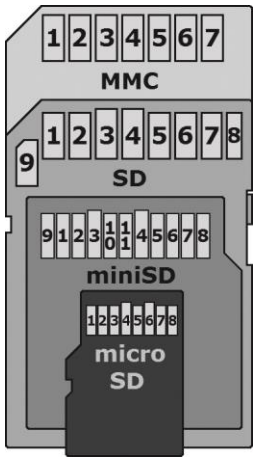
```
  Serial.begin(9600);   Serial.println("EEPROM okuma
ornegi.");   delay(3000);   //adres degeri 256 olana
kadar EEPROM'dan veri oku   for (int addr = 0; addr
<= 320; addr++)   {     //EEPROM'dan addr
adresindeki veriyi oku     int val =
eeprom.read(addr);     //suan okunan EEPROM adres
numarasini yaz
    Serial.print(addr);
     //bosluk birak
     Serial.print("\t");
     //EEPROM'dan okunan veriyi ekrana yaz
Serial.println(val);     delay(200);   }
  Serial.print("Hafizanin sonuna gelindi");
}

void loop()
{

}
```

As you can see, the code used for reading is very similar to the one used for writing. The function eeprom.write() is replaced with eeprom.read() for reading.

Lesson 14: The Usage of SD Card

In our lesson today, we will create a datalogger using the SD card connection on Tinylab to store data collected from various sensors. SD cards, as many of us know, are used for data storage purposes and are commonly found in electronic devices such as mobile phones, computers, and cameras.

SD (Secure Digital) memory cards emerged as a result of efforts by Panasonic and SanDisk in 1999 to enhance the MMC (MultiMedia Card) standard previously developed by Toshiba. They come in three different sizes (standard, mini, and micro) and possess various communication capabilities.

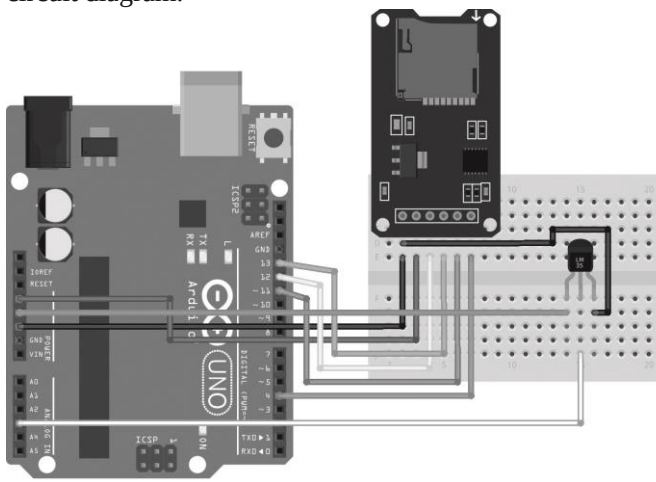


MMC, SD, miniSD ve microSD kartların boyut karşılaştırması.

Although there are faster access methods available, when used with microcontrollers like Arduino, access to an SD card can be achieved via the SPI (Serial Peripheral Interface) bus. Therefore, we need to include the library files SPI.h and SD.h in our code. Additionally, the SD card we use must be formatted with the FAT or FAT32 file system. The connection pins for the SD card are as follows:

SD	microSD	İsim	Giriş/Çıkış	İşlev
1	2	nCS	Giriş	SPI card select [CS] (ters lojik)
2	3	DI	Giriş	SPI seri data girişi [MOSI]
3		VSS	Besleme	Toprak
4	4	VDD	Besleme	Güç
5	5	CLK	Giriş	SPI clock [SCLK]
6	6	VSS	Besleme	Toprak
7	7	DO	Çıkış	SPI seri data çıkışı [MISO]
8	8	NC	-	Kullanılmaz
9	1	NC	-	Kullanılmaz

For those who will build their own circuit with Arduino, here is our circuit diagram:



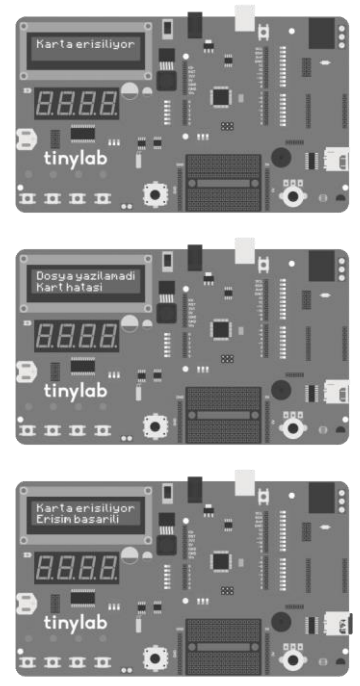
When making connections, it's important to note that if you are using a board like Arduino Leonardo, the SPI connections are not brought out to digital pins. Therefore, you need to use the ICSP (In-Circuit Serial Programming) pins for the connection.

The code:

```
//gerekli kutuphaneler: SPI haberlesme icin SPI.h,
//SD kart icin SD.h,
//I2C haberlesme icin Wire.h,
//saat fonksiyonlari icin Time.h,
//DS1307 RTC entegresi icin DS1307RTC.h,
//I2C LCD icin LiquidTWI2.h
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
#include <LiquidTWI2.h>

//SD kartin chip select (CS) pini ve sicaklik sensoru pin
tanimlamalari #define chipSelect 4
#define LM35_pin A3

//I2C LCD icin tanimlama
LiquidTWI2 lcd(0);

//sicaklik icin baslangic degeri float
temp = 0;

void setup() {   //I2C LCD icin ayarlar
lcd.setMCPType(LTI_TYPE_MCP23008);
lcd.begin(16, 2);
lcd.setBacklight(HIGH);
lcd.print("Karta erisiliyor");   //kart
mevcut degilse veya erisilemiyorsa LCD'ye
hata mesaji yaz   if
(!SD.begin(chipSelect))   {
lcd.setCursor(0, 1);     lcd.print("Kart
hatasi");     return;   }   delay(2500);
//karta erisim mevcutsa LCD'ye mesaj yaz,
daha sonra LCD'yi kapat
lcd.setCursor(0, 1);   lcd.print("Erisim
```

```
basarili");   delay(3000);
lcd.setBackLight(LOW);   lcd.clear(); }

void loop() {   //tm formatinda saat bilgisini okumak icin
bu satiri ekliyoruz   tmElements_t tm;   //SD karta
yazilacak veriyi bir string'de tutmamiz gerekli
  String dataString = "";   //lm35 i oku ve degeri
santigrada cevir   temp = (5.0 * analogRead(LM35_pin)
* 100.0) / 1024;   //string'e sicaklik degerini ekle
dataString += String(temp);

  //RTC'den saat bilgisi alinabiliyorsa
if (RTC.read(tm))
 {
    //SD kartta datalog.txt isimli bir dosya olustur
    File dataFile = SD.open("datalog.txt", FILE_WRITE);
//dosyaya yazilabiliyorsa     if (dataFile)     {
      //saat ve tarih bilgisini dosyaya yaz
//GG//AA/YYYY, SS:DD:ss       if (tm.Day < 10)
dataFile.print("0");       dataFile.print(tm.Day);
dataFile.print("/");       if (tm.Month < 10)
dataFile.print("0");       dataFile.print(tm.Month);
dataFile.print("/");
dataFile.print(tmYearToCalendar(tm.Year));
dataFile.print(",");       if (tm.Hour < 10)
dataFile.print("0");       dataFile.print(tm.Hour);
dataFile.print(":");       if (tm.Minute < 10)
dataFile.print("0");       dataFile.print(tm.Minute);
dataFile.print(":");       if (tm.Second < 10)
dataFile.print("0");       dataFile.print(tm.Second);
//saat ve tarihten sonra sicaklik degerini ekle
dataFile.print(" Sicaklik: ");
dataFile.print(dataString);       dataFile.println(" ¬∞C");
//yazma islemini bitir       dataFile.close();
    }     //dosyaya yazilamiyorsa LCD'de hata mesaji
goster     else     {       lcd.setCursor(0, 0);
```

```
lcd.print("Dosya yazilamadi");      }    }
delay(1000); }
```

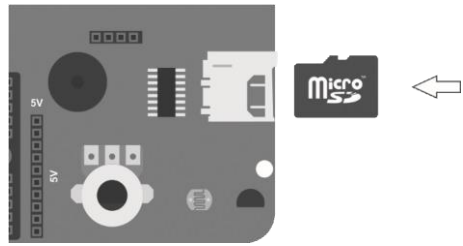


In this tutorial, we'll also be using the Time.h, DS1307RTC.h, and LiquidTWI2.h libraries because we'll be using RTC (Real-Time Clock) and an LCD display in addition to the SD card. We included the Wire.h library because both RTC and the LCD screen use I2C (TWI) communication.

The CS (Chip Select) pin of the SD card on Tinylab is connected to digital pin 4, and the LM35 temperature sensor is connected to pin A3. We define these pins at the beginning of our code.

In the setup function, we initialize the LCD and SD functions. The code here is designed to display error messages on the LCD in case of any connection errors with the SD card or if there's no SD card in the Tinylab slot. This allows us to easily identify any errors that may occur.

To insert the SD card into Tinylab, follow the steps below:



In the loop function, we perform temperature measurement as we did in our analog value reading tutorial. However, in this case, since the data format we'll write to the SD card is defined as a string, we convert the temperature value to a string using the command dataString += String(temp);.

If we're able to read the time information from the RTC (Real-Time Clock) module, we create a file named "datalog.txt" using the command File dataFile = SD.open("datalog.txt", FILE_WRITE); and start writing to this file.

The format of the data we'll store on the SD card will be as follows:

GG/AA/YYYY, SS:DD:ss Sicaklik: xx.xx °C

If you observe the code carefully, you'll notice that we're incorporating the date and time using the same method we used in our previous RTC tutorial. However, in this project, we're using the date and time in the file we write to the SD card instead of displaying it on the serial monitor.

After writing each temperature measurement to the file on the SD card, we add a delay of 1 second (delay(1000)). You can modify this delay period to determine how frequently the recordings are made.

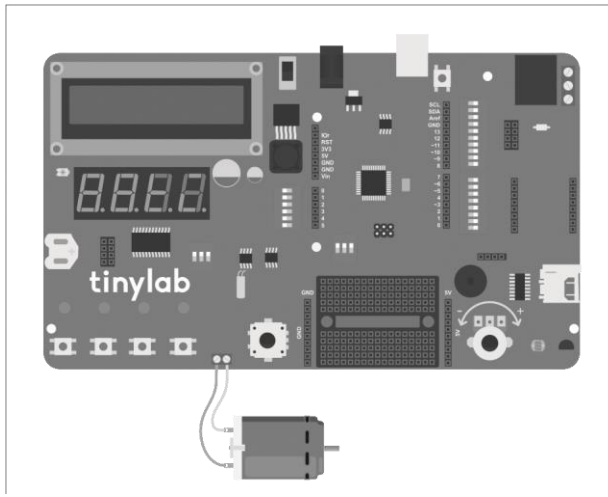
3

Project

In this section, we will learn how to create various projects by using multiple circuits on Tinylab together. These projects include the use of HC-06 Bluetooth, XBee RF, nRF24L01 RF, and ESP8266 WiFi modules, which are part of Tinylab's IoT Kit and Exclusive Kit. For some wireless communication projects, we will need two Arduino boards, one acting as a transmitter and the other as a receiver. Additionally, we will utilize various additional modules, shields, and external components to use our Arduino in various projects. You can obtain all of these products through stores such as Robotistan.com.

7-segment Display Motor Speedometer

In this project, we will use the 7-segment display on Tinylab to display the percentage speed of the motor while controlling the speed of the motor with a potentiometer.



The code:

```
//gerekli kutuphaneler: MAX7219 7-segment surucu icin
LedControl.h
#include <LedControl.h>

//potansiyometre ve motor surucu pin tanimlamalari
#define pot_pin A0
#define motor_pin 5

//DataIn, Clock, Load, surucu sayisi
LedControl lc = LedControl(10, 12, 11, 1);

//baslangic degerleri
int motor_spd = 0; int
percent = 0;

void setup() {   //motor surucunun yer aldigi pini cikis
olarak ayarla   pinMode(motor_pin, OUTPUT);
//gostergeyi calistir   lc.shutdown(0, false);
//gosterge parlakligini 0-15 arasinda ayarla
```

```
lc.setIntensity(0, 8);   //gostergeyi temizle
lc.clearDisplay(0); }

void loop() {   //potanstan gelen 0-1023 arasi degeri PWM
cikis isin 0-255 arasinda olcekle   motor_spd =
analogRead(pot_pin) / 4;   //motor surucu pininden PWM cikis
ver   analogWrite(motor_pin, motor_spd);   //0-255 arasi PWM
degerini yuzde olarak olcekle   int percent = map(motor_spd,
0, 255, 0, 100);   //motor calisma hizini yuzde olarak 7-
segment'e yaz   printToDisplay(percent);   //20ms bekle
delay(20); }

//7-segment gostergeye deger yazmak icin gerekli fonksiyon
void printToDisplay(int value) {   //pozitif sayilari yazma
metodu   if (value >= 0)
  {     //bir basamakli sayilar     if (value < 10)     {
//birler basamagina sayinin 10 ile bolumunden kalani yaz
lc.setDigit(0, 3, (value % 10), false);       //diger
basamaklari bos birak       lc.setRow(0, 2, B00000000);
lc.setRow(0, 1, B00000000);       lc.setRow(0, 0, B00000000);
}     //iki basamakli sayilar     if (value >= 10 && value <
99)     {       //birler basamagina sayinin 10 ile bolumunden
kalani yaz       lc.setDigit(0, 3, (value % 10), false);
//sayiyi 10'a bol, sonucun 10 ile bolumunden kalani onlar
basamagina yaz       lc.setDigit(0, 2, ((value / 10) % 10),
false);       //diger basamaklari bos birak       lc.setRow(0,
1, B00000000);       lc.setRow(0, 0, B00000000);     }
//uc basamakli sayilar     if (value > 100 && value <= 999)
{       //birler basamagina sayinin 10 ile bolumunden kalani
yaz       lc.setDigit(0, 3, (value % 10), false);
//sayiyi 10'a bol, sonucun 10 ile bolumunden kalani onlar
basamagina yaz       lc.setDigit(0, 2, ((value / 10) % 10),
false);       //sayiyi 100'e bol, sonucun 10 ile bolumunden
kalani yuzler basamagina yaz       lc.setDigit(0, 1, ((value /
100) % 10), false);
      //diger basamaklari bos birak
```

```
      lc.setRow(0, 0, B00000000);     }     //dort basamakli
sayilar     if (value >= 1000)     {       //birler basamagina
sayinin 10 ile bolumunden kalani yaz       lc.setDigit(0, 3,
(value % 10), false);       //sayiyi 10'a bol, sonucun 10 ile
bolumunden kalani onlar basamagina yaz       lc.setDigit(0, 2,
((value / 10) % 10), false);       //sayiyi 100'e bol, sonucun
10 ile bolumunden kalani yuzler basamagina yaz
lc.setDigit(0, 1, ((value / 100) % 10), false);       //sayiyi
1000'e bol, sonucun 10 ile bolumunden kalani binler basamagina
yaz       lc.setDigit(0, 0, ((value / 1000) % 10), false);
}
  }   //negatif sayilari yazma metodu   if (value < 0)
{     //bir basamakli sayilar     if (value >= -10)
{       //sayinin mutlak degerinin 10'a bolumunden
kalanini birler basamagina yaz       lc.setDigit(0, 3,
(abs(value) % 10), false);       //bir sonraki basamaga
eksi isareti koy       lc.setRow(0, 2, B00000001);
//diger basamaklari bos birak       lc.setRow(0, 1,
B00000000);       lc.setRow(0, 0, B00000000);     }
//iki basamakli sayilar     if (value >= -100 && value
<= -10)
    {       //sayinin mutlak degerinin 10'a bolumunden
kalanini birler basamagina yaz       lc.setDigit(0, 3,
(abs(value) % 10), false);       //sayinin mutlak degerini
10'a bol, sonucun 10'a bolumunden kalani onlar basamagina
yaz       lc.setDigit(0, 2, ((abs(value) / 10) % 10),
false);       //bir sonraki basamaga eksi isareti koy
lc.setRow(0, 1, B00000001);       //diger basamaklari bos
birak       lc.setRow(0, 0, B00000000);     }     //uc
basamakli sayilar     if (value >= -1000 && value <= -100)
{       //sayinin mutlak degerinin 10'a bolumunden
kalanini birler basamagina yaz       lc.setDigit(0, 3,
(abs(value) % 10), false);       //sayinin mutlak degerini
10'a bol, sonucun 10'a bolumunden kalani onlar basamagina
yaz       lc.setDigit(0, 2, ((abs(value) / 10) % 10),
false);       //sayinin mutlak degerini 100'e bol, sonucun
```

```
 10’a bolumunden kalani yuzler basamagina yaz
 lc.setDigit(0, 1, ((abs(value) / 100) % 10), false);
 //bir sonraki basamaga eksi isareti koy       lc.setRow(0,
 0, B00000001);     }
   }
}
```

At the beginning of our code, we defined the pins for the motor driver and potentiometers connected to the Tinylab. Then, we added the necessary library to use the 7-segment display and defined the pins. We also defined two variables, motor_spd for the motor speed and percent for the motor speed percentage, and initialized them to zero. In the setup function, we set the pin connected to the motor as an output and made the necessary settings for our 7-segment display.

In our loop function, we observe that the variable motor_spd, which determines the speed of the motor, changes based on the value read from the analog port connected to the potentiometer. If you recall from the analog input lesson, the data we read from these inputs comes to us as a value between 0 and 1023. We need to use this value in the analogWrite function to change the motor speed. However, this function allows output from PWM pins based on values between 0 and 255 (for more detailed information, you can refer to the RGB LED lessons). Therefore, we need to scale down the value ranging from 0 to 1023 to fit within the range of 0 to 255. We can achieve this by either dividing it by 4 or using the map function. In this code, both methods are used as examples.

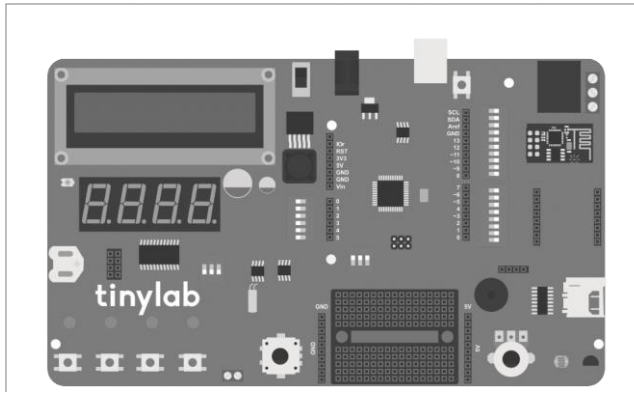
1023, being approximately 4 times 255 (since there are actually 256 steps between 0 and 255, and 1024 steps between 0 and 1023), a simple division operation is sufficient for us. However, if we want to map the range to another range (for example, mapping the range from 0 to 255 to -100 to +100), it is more convenient to use the map function. The usage of this function is as follows:

```
map(olceklenecek_deger, degerin_baslangici, degerin_bitisi,
istenilen_altlimit, istenilen_ustlimit)
```

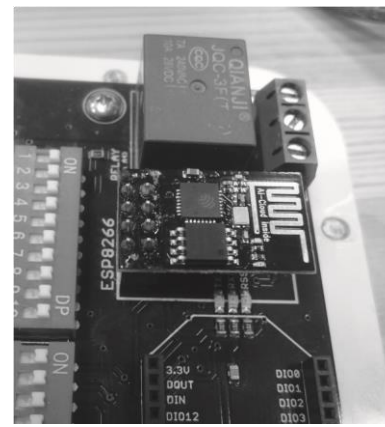
In this application, since we want to view the motor speed as a percentage, we used the map function in the following way: map(motor_spd, 0, 255, 0, 100). The printToDisplay function at the bottom of the code allows us to easily write values to the 7-segment display in the rest of the code by using printToDisplay(numeric_value) format. If you want to examine the functions here, you can refer to our 7-segment display lesson.

The Usage of ESP8266 WiFi Module

The ESP8266 is a WiFi module that is widely popular in Arduino projects due to its ease of use and particularly its affordability. Despite hosting its own microcontroller, it can operate without the need for an additional microcontroller like Arduino. Moreover, it is frequently preferred in many Internet of Things (IoT) projects due to its support for the AT command set.
In this project, I will explain the necessary steps to establish communication between the ESP8266 module and our Tinylab board, as well as to provide internet connectivity.



The ESP8266 establishes a connection with the Tinylab using the UART communication interface. Its usage is straightforward. By connecting the module to the ESP8266 socket on the Tinylab as shown in the image below, we can start using it immediately.

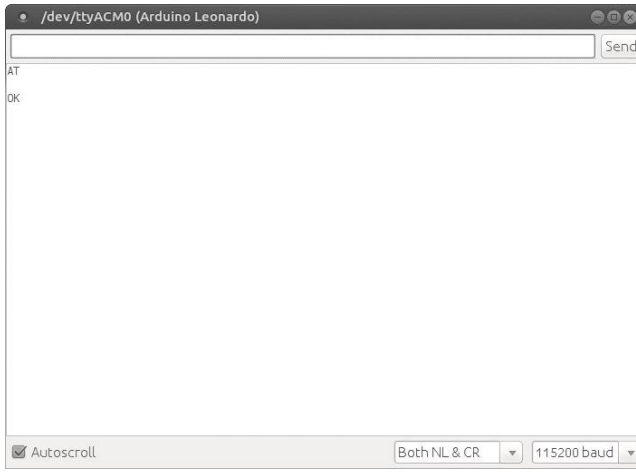


After properly inserting our module into its socket, we load the following code onto our Tinylab for initial setup:

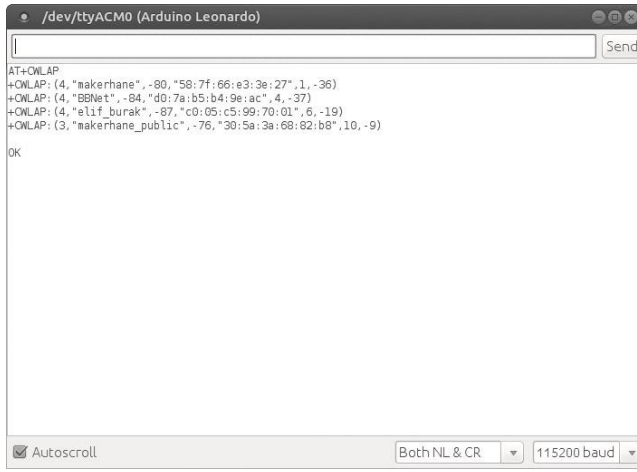
```
void setup()  {
Serial.begin(9600);
  Serial1.begin(115200);   }   void
loop()         {                  if
(Serial1.available()) {         int
inByte       =       Serial1.read();
Serial.write(inByte);
```

```
    }    if (Serial.available())
{      int inByte =
Serial.read();
Serial1.write(inByte);
  }
}
```

This code serves as a bridge between the virtual serial port used by our Tinylab for communication with the computer via USB and the hardware port connected to the ESP8266 module. To confirm that our ESP8266 module is functioning correctly, we open the serial port monitor and send the AT command. If you receive the "OK" response as seen in the screenshot below, it means your module is working fine.



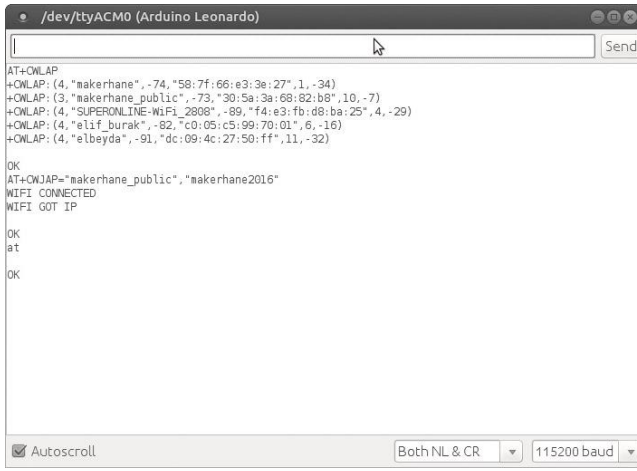
To list the WiFi access points in the coverage area, we issue the command "AT+CWLAP".



To connect to a wireless access point listed here, we use the command:

AT+CWJAP=”kablosuz_ag_ismi”,”sifre”

When the connection is established, the module notifies us as follows:



disconnect from the wireless access point, we use the command: **AT+CWQAP**

In addition, here are some other commands:

AT+GMR -> Shows the firmware version.

AT+CIFSR -> Displays the IP address obtained by the module

AT+RST - > Resets the module

For all AT commands of the ESP8266 module, you can refer to the PDF file available on the GitHub link.

https://github.com/keremizgol/tinylab-kitap/blob/master/kodlar/projeler/esp8266_bridge/4A-ESP8266__AT_Instruction_Set__EN_v0.30.pdf

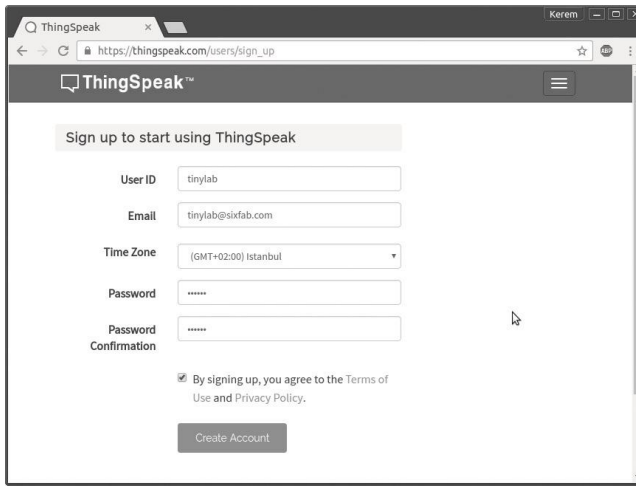
Sending Sensor Data to ThingSpeak with ESP8266

In this project, we will learn how to send sensor data to the ThingSpeak website using the sensors on the Tinylab and the ESP8266 WiFi module.

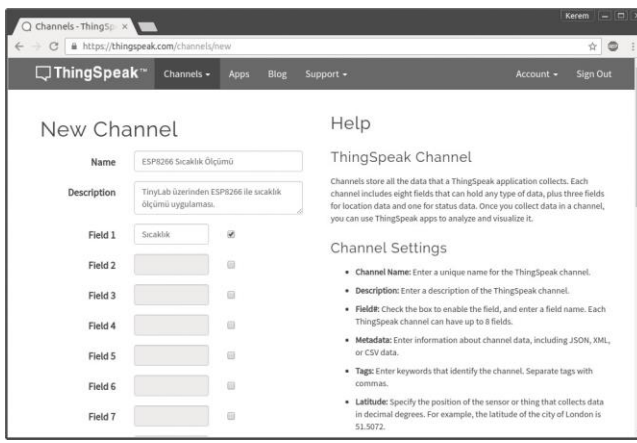
ThingSpeak is a platform designed for IoT devices. It offers features such as real-time data collection and storage, analysis using MATLAB, task scheduling, and more, making it an ideal choice for our project.

Before starting our project, you need to follow the instructions in the "Using the ESP8266 WiFi Module" project to pair your module with a wireless access point and connect it to the internet. Once you've done that, you can create a new account on ThingSpeak.com and begin this project. It's important to enter the correct "Time Zone" setting to ensure

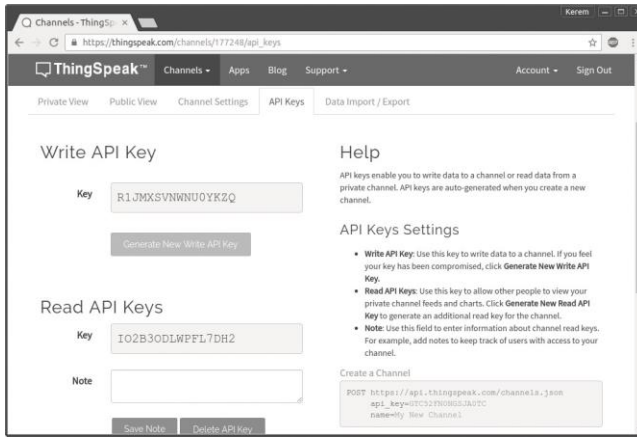
that the sensor data is recorded with the correct timestamp.



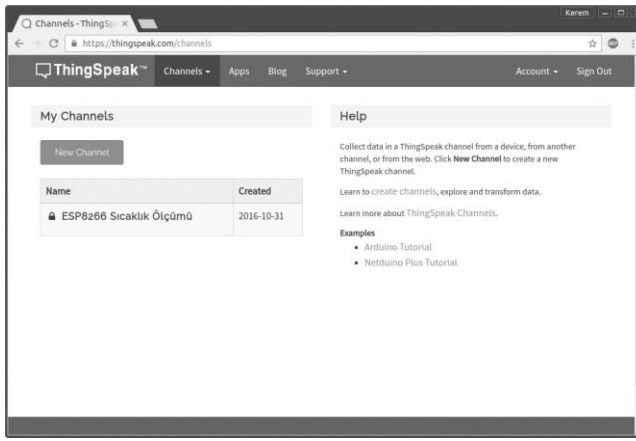
After creating our account, we need to create a new channel by clicking the "New channel" button under the "Channels" tab. You can configure the channel settings as shown in the screenshot below. You don't need to write it exactly as shown, but make sure that "Field 1" option is enabled.



After configuring the channel settings, we will need the "Write API Key" under the "API Keys" tab. We need to write this key into the line "String api_key = "xxxxxxxxxxxxxxx";" in our code.



After completing these steps, when we log into our ThingSpeak account and click on "My Account" in the top right corner, we should be able to see the channel we created.



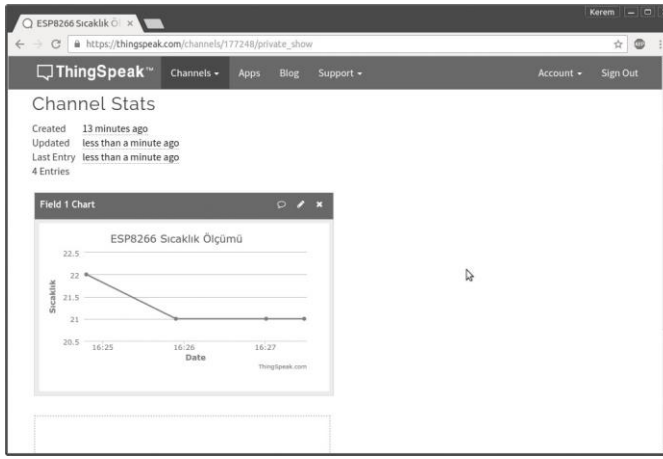
If we have reached this point smoothly, now it's time to upload the code to Tinylab.

```
//thingspeak.com ip adresi
#define IP "184.106.153.149"
//sicaklik sensoru pin baglantisi
#define lm35_pin A3
```

```
//Thingspeak'ten alacaginiz Write API Key
//kodu yuklemeden once degistirmeyi unutmayiniz
String api_key = "xxxxxxxxxxxxxxxx";


void setup()
{
  //USB-seri baglantisi
  Serial.begin(9600);
  //Tinylab-ESP8266 baglantisi
  Serial1.begin(115200);
delay(5000); }

void loop() {   //sicaklik sensorunu oku   int temp = (5.0
* analogRead(lm35_pin) * 100.0) / 1024;   //USB-seri
portuna sicakligi yaz
  Serial.println(temp);   //Thingspeak'e
sicaklik bilgisini yolla
send_temp(temp);   //1 dk bekle
delay(60000); }


void send_temp(int temp) {   //Thingspeak sunucusuna 80 numarali
porttan            TCP            iletisimini            baslat
Serial.println(String("AT+CIPSTART=\"TCP\",\"") + IP +
"\",80");
  Serial1.println(String("AT+CIPSTART=\"TCP\",\"") + IP +
"\",80");   delay(1000);   if (Serial1.find("Error"))
//baglanti hatasi durumunda seri monitorden bildir   {
Serial.println("AT+CIPSTART Error");     return;   }

  //sicaklik bilgisi ve Write API Key ile gonderilecek komutu
String tipinde bir degiskende tut
  String cmd = "GET /update?key=";
  cmd += api_key;
cmd += "&field1=";
cmd += (int(temp));
```

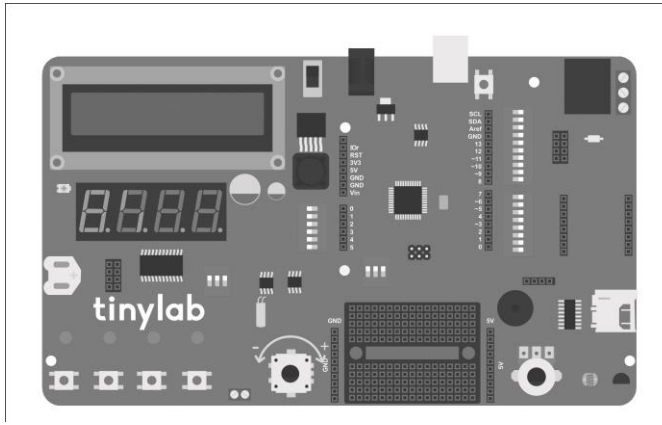
```
cmd += "\r\n\r\n";
delay(3000);

  //hazirlanan komutun boyutunu cmd.lenght() ile bul
  Serial.print("AT+CIPSEND=");
  Serial1.print("AT+CIPSEND=");
  Serial.println(cmd.length() + 2);
Serial1.println(cmd.length() + 2);

 delay(1000);

  //komut gonderilmeye hazir ise cmd String'ini gonder
if (Serial1.find(">"))
 {
   Serial.print(cmd);
   Serial1.print(cmd);
    Serial.print("\r\n\r\n");
    Serial1.print("\r\n\r\n");
  }   //komut gonderilmeye hazir degil ise baglantiyi
kapat   else   {
    Serial.println("AT+CIPCLOSE");
    Serial1.println("AT+CIPCLOSE");
 }
}
```

If everything went smoothly, don't forget to enter the "Write API Key" of the channel we created into the api_key part of the code. If there are no issues, we should be able to see the temperature values changing on ThingSpeak.


ESP8266 Sıcaklık Öl
Kerem
https://thingspeak.com/channels/177248/private_show
ThingSpeak™
Channels
Apps
Blog
Support
Account
Sign Out
Channel Stats
Created 13 minutes ago
Updated less than a minute ago
Last Entry less than a minute ago
4 Entries
Field 1 Chart
ESP8266 Sıcaklık Ölçümü
Sıcaklık
22.5
22
21.5
21
20.5
16:25
16:26
16:27
Date
ThingSpeak.com

7-segment ile rotary encoder

In this project, we will use the rotary encoder together with our 7-segment display, which we have used before.




```
//gerekli kutuphaneler: MAX7219 7-segment surucu icin
//LedControl.h
#include <LedControl.h>


//rotary enkoder icin baglanti pinleri
#define pinA 6
#define pinB 7

//DataIn, Clock, Load, surucu sayisi
LedControl lc = LedControl(10, 12, 11, 1);

//baslangic degerleri
int encoderPos = 0;
bool pinALast = LOW;
bool n = LOW;

void setup() {
  //enkoder pinlerini giris olarak ayarla, pull-up
direnclerini aktiflestir   pinMode (pinA, INPUT_PULLUP);
```

```
pinMode (pinB, INPUT_PULLUP);   //gostergeyi calistir
lc.shutdown(0, false);   //gosterge parlakligini 0-15 arasinda
ayarla   lc.setIntensity(0, 8);   //gostergeyi temizle
lc.clearDisplay(0); }

void loop() {   //enkoderin A pinini oku   n =
digitalRead(pinA);   //A pini 0 dan 1 e gectiyse   if
((pinALast == LOW) && (n == HIGH))   {     //B pini 0
daysa     if (digitalRead(pinB) == LOW)
encoderPos++; //enkoder degerini arttir     else
encoderPos--; //eknoder degerini azalt     //enkoder
degerini 7-segment gostergeye yaz
printToDisplay(encoderPos);   }   pinALast = n; //A
pininin en son durumunu n'e esitle   //enkoder butonuna
basildiginda enkoder degerini sifirla   if
(analogRead(A5) >= 80 && analogRead(A5) <= 100)   {
encoderPos = 0;     printToDisplay(encoderPos);   }
}
//7-segment gostergeye deger yazmak icin gerekli fonksiyon
void printToDisplay(int value) {   //pozitif sayilari yazma
metodu   if (value >= 0)   {     //bir basamakli sayilar
if (value < 10)     {       //birler basamagina sayinin 10 ile
bolumunden kalani yaz       lc.setDigit(0, 3, (value % 10),
false);       //diger basamaklari bos birak       lc.setRow(0,
2, B00000000);       lc.setRow(0, 1, B00000000);
lc.setRow(0, 0, B00000000);     }     //iki basamakli sayilar
if (value >= 10 && value < 99)     {       //birler basamagina
sayinin 10 ile bolumunden kalani yaz       lc.setDigit(0, 3,
(value % 10), false);       //sayiyi 10'a bol, sonucun 10 ile
bolumunden kalani onlar basamagina yaz       lc.setDigit(0, 2,
((value / 10) % 10), false);       //diger basamaklari bos
birak       lc.setRow(0, 1, B00000000);       lc.setRow(0, 0,
B00000000);     }     //uc basamakli sayilar     if (value >
100 && value <= 999)     {       //birler basamagina sayinin
10 ile bolumunden kalani yaz       lc.setDigit(0, 3, (value %
10), false);
```

```
      //sayiyi 10'a bol, sonucun 10 ile bolumunden kalani onlar
basamagina yaz      lc.setDigit(0, 2, ((value / 10) % 10), false);
//sayiyi 100'e bol, sonucun 10 ile bolumunden kalani yuzler
basamagina yaz          lc.setDigit(0, 1, ((value / 100) % 10),
false);       //diger basamaklari bos birak       lc.setRow(0, 0,
B00000000);      }      //dort basamakli sayilar      if (value >=
1000)      {        //birler basamagina sayinin 10 ile bolumunden
kalani yaz            lc.setDigit(0, 3, (value % 10), false);
//sayiyi 10'a bol, sonucun 10 ile bolumunden kalani onlar
basamagina yaz      lc.setDigit(0, 2, ((value / 10) % 10), false);
//sayiyi 100'e bol, sonucun 10 ile bolumunden kalani yuzler
basamagina yaz          lc.setDigit(0, 1, ((value / 100) % 10),
false);          //sayiyi 1000'e bol, sonucun 10 ile bolumunden
kalani binler basamagina yaz        lc.setDigit(0, 0, ((value /
1000) % 10), false);      }
  }   //negatif sayilari yazma metodu   if (value < 0)
{      //bir basamakli sayilar      if (value >= -10)
{        //sayinin mutlak degerinin 10'a bolumunden
kalanini birler basamagina yaz       lc.setDigit(0, 3,
(abs(value) % 10), false);       //bir sonraki basamaga
eksi isareti koy       lc.setRow(0, 2, B00000001);
      //diger basamaklari bos birak       lc.setRow(0, 1,
B00000000);       lc.setRow(0, 0, B00000000);      }      //iki
basamakli sayilar      if (value >= -100 && value <= -10)      {
//sayinin mutlak degerinin 10'a bolumunden kalanini birler
basamagina yaz       lc.setDigit(0, 3, (abs(value) % 10),
false);       //sayinin mutlak degerini 10'a bol, sonucun 10'a
bolumunden kalani onlar basamagina yaz       lc.setDigit(0, 2,
((abs(value) / 10) % 10), false);       //bir sonraki basamaga
eksi isareti koy       lc.setRow(0, 1, B00000001);
//diger basamaklari bos birak       lc.setRow(0, 0,
B00000000);
   }
    //uc basamakli sayilar      if (value >= -1000 && value <= -
100)      {       //sayinin mutlak degerinin 10'a bolumunden
kalanini birler basamagina yaz       lc.setDigit(0, 3,
(abs(value) % 10), false);       //sayinin mutlak degerini 10'a
bol, sonucun 10'a bolumunden kalani onlar basamagina yaz
```

```
lc.setDigit(0, 2, ((abs(value) / 10) % 10), false);
//sayinin mutlak degerini 100'e bol, sonucun 10'a bolumunden
kalani yuzler basamagina yaz       lc.setDigit(0, 1,
((abs(value) / 100) % 10), false);       //bir sonraki basamaga
eksi isareti koy       lc.setRow(0, 0, B00000001);     }
  }

}
```

Our code is actually very simple. The setup function consists of a combination of the 7-segment display lesson and the rotary encoder lesson. The loop function is almost the same as the one in the rotary encoder lesson, with the difference being that the encoder position is displayed on the 7-segment display instead of the serial monitor.
To easily display values on the 7-segment display, we use the printToDisplay() function. In this function, we first check if the incoming parameter (the int value parameter inside the parentheses) is negative or positive. If the incoming value is negative (i.e., if the encoder is turned counterclockwise), a minus sign is added to the beginning of the value. Remembering how each LED of the 7-segment display is configured from the 7-segment display lesson, we know that we need to enter the value B00000001 for the minus sign. Additionally, for the value to be displayed properly, each digit of our 7-segment display needs to be sent to the display individually. For this, we use the modulo operator (%) and a simple division operation. For negative numbers, we use the abs() function to obtain the absolute value of the number.
We can easily copy the printToDisplay function included in this project to any other project we want, allowing us to easily write integer values to the 7-segment display on the Tinylab. Additionally, we can control variables such as motor speed and servo motor position with the rotary encoder.

The Usage of Arduino Without Delay

When writing an Arduino program, we have always used the delay() function for any waiting process. When we use this function, the microcontroller of our Arduino waits without performing any operation for the specified duration, allowing us to obtain the necessary waiting times, for example, to blink an LED. However, what if we want to do multiple tasks at the same time?

In this case, we need to think a bit more carefully when writing code. We will instruct Arduino to perform the required tasks at certain time intervals while continuously monitoring the time. For example, instead of sitting in front of the washing machine waiting for it to finish after filling it up and starting it, we can periodically check whether the washing is finished. Meanwhile, we can engage in other tasks.

The millis() function in Arduino allows us to obtain the running time of the program in milliseconds. As long as there is no power outage, this function resets to zero after counting for approximately 50 days. By using this function, we can note the time of the tasks performed by our program and check whether the millis() function has increased by the repetition period to perform the same waiting operation. Let's try writing the LED blinking, which is essential in Arduino programs, using this method.

```
//LED'in bagli oldugu pin
#define ledPin 13

//baslangic degerleri ve bekleme suresi
bool ledDurum = LOW; unsigned long
oncekiZaman = 0; const long aralik =
1000;

void setup() {   //LED pinini
cikis olarak ayarla
pinMode(ledPin, OUTPUT); }

void loop()
{
  //suanki zamani hafizada tut
unsigned long simdikiZaman = millis();

  //suanki zaman ile bir onceki zamanin farki bekleme suresi
kadar ise   if (simdikiZaman - oncekiZaman >= aralik)   {
    //zaman bilgisini sifirla
oncekiZaman = simdikiZaman;
//ledDurum'u tersine cevir
```

```
ledDurum = !ledDurum;   }
//ledDurum degiskenini ledPin'e yaz
digitalWrite(ledPin, ledDurum); }
```

To explain the code, first, as in every lesson, we make our declarations. We use variables like ledPin for the pin to which our LED is connected, ledDurum for the current state of the LED (on or off), oncekiZaman to keep track of the previous time the LED was turned on, and aralik for the blinking interval of the LED.

NOTE: The variables oncekiZaman and aralik are defined as unsigned long and const long, respectively, because they represent time. unsigned long is used for unsigned decimal numbers, while const long is used for constant (unchangeable) decimal numbers. In the setup function, we set the ledPin as an output.

The loop function may seem a bit complicated, but what we are doing is quite simple: We store the current millis value, which is essentially the Arduino's clock, in the variable simdikiZaman. If the time elapsed since the last time the LED was turned on or off is equal to or greater than the desired blinking interval (specified by the aralik variable), we enter the if statement. Inside the if statement, we reset the time (set it equal to the current time) and change the state of the LED. After this change, the digitalWrite(ledPin, ledDurum); command outside the if statement controls whether the LED turns on or off based on the value of the ledDurum variable.

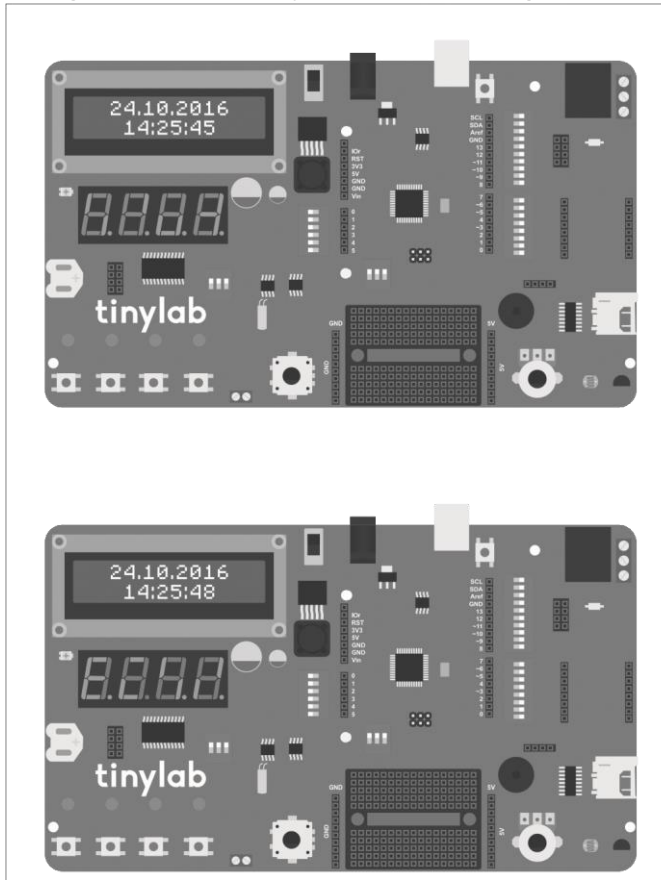
We can easily modify this to make two LEDs blink at different speeds using the following code:

```
//LED'lerin bagli oldugu pinler
#define led1Pin 13
#define led2Pin 12
```

```
//baslangic degerleri ve bekleme sureleri
bool led1Durum = LOW; bool led2Durum =
LOW; unsigned long oncekiZaman1 = 0;
unsigned long oncekiZaman2 = 0; const
long aralik1 = 500; const long aralik2 =
300;

void setup()
{   //LED pinlerini cikis olarak ayarla
pinMode(led1Pin, OUTPUT);
pinMode(led2Pin, OUTPUT); }

void loop() {
  //suanki zamani hafizada tut   unsigned
long simdikiZaman = millis();

  //suanki zaman ile bir onceki zamanin farki birinci LED'in
bekleme suresi kadar ise   if(simdikiZaman - oncekiZaman1 >=
aralik1)   {
    //zaman bilgisini sifirla
oncekiZaman1 = simdikiZaman;
//led1Durum'u tersine cevir
led1Durum = !led1Durum;   }

  //suanki zaman ile bir onceki zamanin farki ikinci LED'in
bekleme suresi kadar ise   if(simdikiZaman - oncekiZaman2 >=
aralik2)   {
    //zaman bilgisini sifirla
oncekiZaman2 = simdikiZaman;
//led2Durum'u tersine cevir     led2Durum =
!led2Durum;   }   //LED durum
degiskenlerini LED'lere yaz
digitalWrite(led1Pin, led1Durum);
digitalWrite(led2Pin, led2Durum);
}
```

Thermometer and Calender Clock

In this project, we will create a clock application using the RTC, LCD, temperature sensor, and 7-segment display on the Tinylab.
Before uploading the code for the project, I recommend reviewing the lessons related to the components used in the project. This project serves as a good example of using multiple components together on the Tinylab and avoiding the use of the delay function for waiting




```
//gerekli kutuphaneler: I2C haberlesme icin Wire.h
//saat fonksiyonlari icin Time.h,
//DS1307 RTC entegresi icin DS1307RTC.h,
```

```
//gerekli kutuphaneler: MAX7219 7-segment surucu icin
LedControl.h
//I2C LCD icin LiquidTWI2.h
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
#include <LedControl.h>
#include <LiquidTWI2.h>

//sicaklik sensoru pini
#define LM35_PIN A3

//I2C LCD icin tanimlama
LiquidTWI2 lcd(0x20);
//DataIn, Clock, Load, surucu sayisi
LedControl lc = LedControl(10, 12, 11, 1);

//delay kullanmadan bekleme fonksiyonu icin degiskenler bool
state = LOW; int previousMillis = 0; int interval = 2500;

//7-segment gosterge icin degiskenler int
temp = 0; int temp_first_digit = 0; int
temp_second_digit = 0; int hour_first_digit =
0; int hour_second_digit = 0; int
minute_first_digit = 0; int
minute_second_digit = 0;

void setup() {   //gostergeyi
calistir   lc.shutdown(0,
false);   //gosterge
parlakligini 0-15 arasinda
ayarla   lc.setIntensity(0, 8);
//gostergeyi temizle
lc.clearDisplay(0);   //I2C LCD
icin ayar
lcd.setMCPType(LTI_TYPE_MCP23008
);   //16 sutun, 2 satir
lcd.begin(16, 2);   //arka isigi
ac   lcd.setBacklight(HIGH);
delay(300); }

void loop() {
```

```
  //suanki zamani hafizada tut   int currentMillis = millis();
//lm35 i oku ve degeri santigrada cevir   temp = (5.0 *
analogRead(LM35_PIN) * 100.0) / 1024;   //lcd ekrana tarih ve
saati yaz   lcd_print();   //suanki zaman ile bir onceki
zamanin farki bekleme suresi kadar ise   if (currentMillis -
previousMillis > interval)   {
    //zaman bilgisini sifirla
previousMillis = currentMillis;
//state degiskeni 0 ise     if (state ==
LOW)     {       //7-segment gostergeye
saati yaz       led_clock();
//state degiskenini 1 yap       state =
HIGH;     }     //state degiskeni 0
degil ise     else     {
      //7-segment gostergeye sicaklik bilgisini yaz
led_temp();       //state degiskenini 0 yap
state = LOW;     }
  }


}

//7-segment gostergeye saati yazma fonksiyonu void led_clock()
{   //tm formatinda saat bilgisini okumak icin bu satiri
ekliyoruz   tmElements_t tm;   if (RTC.read(tm))//RTC
entegresinden tm formatinda okunabiliyorsa   {     //saat
degerini basamaklarina ayir     convertTwoDigits(tm.Hour,
&hour_first_digit, &hour_second_ digit);     //7-segment
gostergeye saati basamak basamak yaz     lc.setDigit(0, 0,
hour_first_digit, false);     lc.setDigit(0, 1,
hour_second_digit, true);     //dakika degerini basamaklarina
ayir     convertTwoDigits(tm.Minute, &minute_first_digit,
&minute_ second_digit);     //7-segment gostergeye dakikayi
basamak basamak yaz     lc.setDigit(0, 2, minute_first_digit,
false);     lc.setDigit(0, 3, minute_second_digit, false);   }
else //RTC entegresi okunamiyorsa tum hanelere 8 yaz   {
lc.setDigit(0, 0, 8, true);     lc.setDigit(0, 1, 8, true);
lc.setDigit(0, 2, 8, true);     lc.setDigit(0, 3, 8, true);
  }
}

//7-segment gostergeye sicaklik yazma fonksiyonu void
led_temp() {   //sicaklik degerini basamaklarina ayir
convertTwoDigits(temp, &temp_first_digit, &temp_second_digit);
//7-segment gostergenin ilk iki hanesine sicakligi basamak
basamak yaz   lc.setDigit(0, 0, temp_first_digit, false);
```

```
lc.setDigit(0, 1, temp_second_digit, false);   //7-segment
gostergenin son iki hanesine santigrad derece sembolu yaz
lc.setRow(0, 2, B01100011); //derece sembolu   lc.setRow(0, 3,
B01001110); //C harfi
}

//iki basamakli sayiyi basamaklarina ayirma fonksiyonu void
convertTwoDigits(int number, int *first, int *second ) {
//sayi 10'dan kucukse tek basamaklidir, ilk haneyi 0 yap
if (number >= 0 && number < 10)   {
    *first = 0;
    *second = number;
  }   //sayi 10'dan
buyukse   else   {
    //onlar basamagini sayinin 1/10'una esitle
    *first = number / 10;
    //birler basamagini sayinin 10'dan bolumune kalanina esitle
    *second = number % 10;
  }
}

//RTC entegresini okuma fonksiyonu
void rtc_read()
{
  //tm formatinda saat bilgisini okumak icin bu satiri
ekliyoruz   tmElements_t tm;   //RTC entegresi okunamiyorsa
ekrana hata mesaji yaz   if (RTC.chipPresent())
lcd.print("RTC Okuma Hatasi!"); }

//LCD ekrana tarih ve saat yazma fonksiyonu
//GG.AA.YYYY //SS:DD:ss void lcd_print() {   //tm formatinda
saat bilgisini okumak icin bu satiri ekliyoruz   tmElements_t
tm;   //RTC entegresinden tm formatinda okunabiliyorsa   if
(RTC.read(tm))   {     //imleci ekranin birinci satir dorduncu
sutuna getir     lcd.setCursor(3, 0);     if (tm.Day < 10)
//gun tek haneliyse basina 0 koy       lcd.print("0");
//gun bilgisini yaz     lcd.print(tm.Day);     //gun ile ay
ayiraci koy     lcd.print(".");     if (tm.Month < 10) //ay tek
haneliyse basina 0 koy       lcd.print("0");     //ay bilgisini
yaz     lcd.print(tm.Month);     //ay ile yil ayiraci koy
lcd.print(".");     //yili dort haneli olarak yaz
lcd.print(tmYearToCalendar(tm.Year));     //imleci ikinci satir
besinci sutuna getir     lcd.setCursor(4, 1);     if (tm.Hour <
```

```
10) //saat tek haneliyse basina 0 koy       lcd.print("0");
//saat bilgisini yaz     lcd.print(tm.Hour);     //saat ile
dakika ayiraci koy     lcd.print(":");     if (tm.Minute < 10)
//dakika tek haneliyse basina 0 koy       lcd.print("0");
//dakika bilgisini yaz     lcd.print(tm.Minute);     //dakika
ile saniye ayiraci koy     lcd.print(":");     if (tm.Second <
10) //saniye tek haneliyse basina 0 koy       lcd.print("0");
//saniye bilgisini yaz     lcd.print(tm.Second);   }   else
//RTC okunamiyorsa ekrana hata mesaji yaz   {     if
(RTC.chipPresent())     {       lcd.setCursor(0, 0);
lcd.print("RTC Okuma Hatasi");     }
  }

}
```

Relay Control with RC522 RFID Module

RFID, generally used for recognizing objects using radio waves, is a technology commonly encountered in our daily lives, such as in public transportation tickets, turnstiles at workplaces and schools. In microcontroller projects, the RC522 module is often preferred for RFID usage. Along with the module, RFID tags that can be used in keychain or card form are provided.

The cards we use have a unique number called UID. This number is different for each card. When we bring our card or keychain close to the reader, this number is read and processed. In this application, we will first save the UIDs of our cards to the internal EEPROM on the Tinylab and then compare the UID of the card we read with the UID values in memory to perform an action.

The RC522 module has an SPI connection. SPI (Serial Peripheral Interface) is a synchronous serial communication interface formed by the initials of the words Serial Peripheral Interface. It was developed by Motorola in the late 1980s and is widely used in the electronics industry. Many devices such as SD cards, LCD screens, and wireless communication modules support SPI.

We connect the SCK, MOSI, MISO, and RST pins of the RC522 module to the SPI connectors located just above the breadboard on the Tinylab. We connect the 3.3V pin to the 3.3V on the Arduino part, the GND to any GND pin, and finally the SDA pin of the RC522 to Arduino

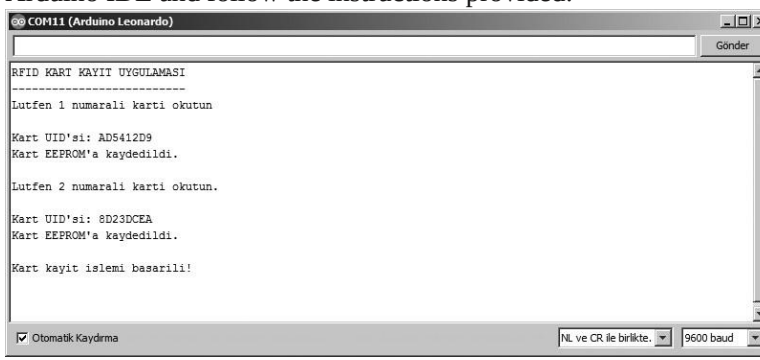
pin 10. Since Arduino pin 10 is also connected to L4 on the Tinylab, we do not forget to close the switch next to it.

Our code consists of two parts: saving the UID information of the cards we have to the EEPROM and controlling the relay by reading the stored UIDs from the EEPROM. As the first step, we upload the necessary code to our Tinylab to save the UIDs of the cards to EEPROM.

```
#include <SPI.h>
#include <MFRC522.h>
#include <EEPROM.h>

#define RST_PIN 9 #define SS_PIN
10

byte readCard[4]; int successRead; MFRC522

mfrc522(SS_PIN, RST_PIN);

MFRC522::MIFARE_Key key;

void setup() {   Serial.begin(9600);   while (!Serial);
SPI.begin();   mfrc522.PCD_Init();   Serial.println("RFID
KART KAYIT UYGULAMASI");
  Serial.println("--------------------------");
  Serial.println("Lutfen 1 numarali karti okutun");
Serial.println();   do {     successRead = getID();   }
while (!successRead);   for ( int i = 0; i < mfrc522.uid.size;
i++ )   {
    EEPROM.write(i, readCard[i] );
  }
  Serial.println("Kart EEPROM'a kaydedildi.");
  Serial.println();
  Serial.println("Lutfen    2    numarali    karti    okutun.");
Serial.println();   do {     successRead = getID();   }   while
(!successRead);   for ( int i = 0; i < mfrc522.uid.size; i++ )
{
    EEPROM.write(i + 4, readCard[i] );
```

```
  }
   Serial.println("Kart EEPROM'a kaydedildi.");
  Serial.println();
   Serial.println("Kart kayit islemi basarili!"); }

void loop() { }

int getID() {   if ( ! mfrc522.PICC_IsNewCardPresent())
{     return 0;   }   if ( !
mfrc522.PICC_ReadCardSerial()) {     return 0;   }
  Serial.print("Kart UID'si: ");     for (int i = 0; i <
mfrc522.uid.size; i++) {  //     readCard[i] =
mfrc522.uid.uidByte[i];     Serial.print(readCard[i], HEX);
  }   Serial.println("");
mfrc522.PICC_HaltA();   return 1;
}
```

After uploading the code, we open the serial port monitor from the Arduino IDE and follow the instructions provided:



After following the instructions displayed on the serial port monitor, we upload the code to control the relay with the RC522 to our Tinylab:

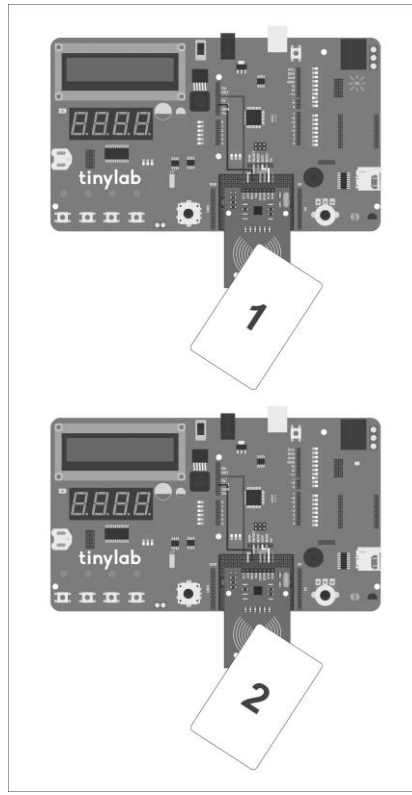
```
#include <SPI.h>
#include <MFRC522.h>
#include <EEPROM.h>

#define RST_PIN 9
#define SS_PIN 10
```

```
#define relayPin A4

MFRC522 mfrc522(SS_PIN, RST_PIN);

String lastRfid = "";
String kart1 = "";
String kart2 = "";

MFRC522::MIFARE_Key key;

void setup()
{
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();   pinMode(relayPin, OUTPUT);
Serial.println("RFID KART OKUMA UYGULAMASI");
  Serial.println("--------------------------");
Serial.println();   readEEPROM(); }

void loop() {   if ( !
mfrc522.PICC_IsNewCardPresent())   {     return;
}   if ( ! mfrc522.PICC_ReadCardSerial())   {
return;   }
  String rfid = "";   for (byte i = 0; i < mfrc522.uid.size;
i++)   {     rfid += mfrc522.uid.uidByte[i] < 0x10 ? " 0" : "
";     rfid += String(mfrc522.uid.uidByte[i], HEX);
  }   rfid.trim();
rfid.toUpperCase();

  if (rfid == lastRfid)
return;   lastRfid = rfid;

  Serial.print("Kart 1: ");
  Serial.println(kart1);
  Serial.print("Kart 2: ");
  Serial.println(kart2);
  Serial.print("Okunan: ");
  Serial.println(rfid);
```

```
  Serial.println();

   if (rfid == kart1)
  {
     digitalWrite(relayPin, HIGH);
 Serial.println("Role kesimde.");   }   if (rfid ==
 kart2)
  {
     digitalWrite(relayPin, LOW);     Serial.println("Role
 iletimde.");
  }   Serial.println();
delay(200); }

void readEEPROM() {   for (int i = 0 ; i < 4 ; i++)   {     kart1
+=  EEPROM.read(i)  <  0x10  ?  " 0"  :  "  ";          kart1  +=
String(EEPROM.read(i), HEX);   }

   for (int i = 4 ; i < 8 ; i++)   {     kart2 += EEPROM.read(i)
 < 0x10 ? " 0" : " ";     kart2 += String(EEPROM.read(i), HEX);
 }     kart1.trim();   kart1.toUpperCase();   kart2.trim();
 kart2.toUpperCase();
}
```

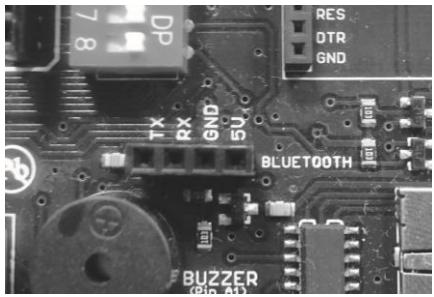
At this stage, when we read the card numbered 1, our relay will turn on, and when we read the card numbered 2, it will turn off.


COM11 (Arduino Leonardo)
Gönder
Kart 1: AD 54 12 D9
Kart 2: 8D 23 DC EA
Okunan: 8D 23 DC EA
Role iletimde.
Kart 1: AD 54 12 D9
Kart 2: 8D 23 DC EA
Okunan: AD 54 12 D9
Role kesimde.
Otomatik Kaydırma
NL ve CR ile birlikte.
9600 baud

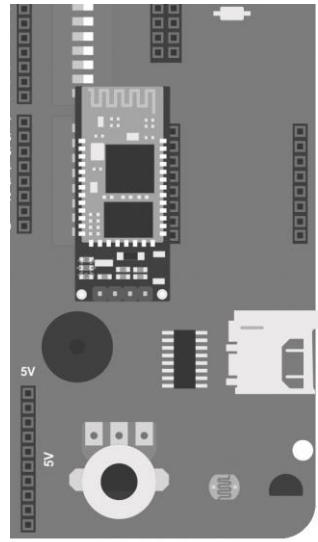

tinylab
1
tinylab
2

Controlling LED via Smart Device Using HC-06 Bluetooth Module

Bluetooth, a technology present in almost all devices from our smartphones to our headphones, provides wireless communication capability. In order to add Bluetooth connectivity support to our projects with Arduino, various modules are available on the market. Tinylab has a compatible connection port for popular Bluetooth modules such as HC-05 and HC-06:



HC06 modül konektörü, Tinylab kartı üzerinde buzzer'ın hemen üst kısmında yer alır.



In this project, we will use the HC-06 module. It is also possible to use the HC-05 instead of HC-06, but the configuration procedure works differently. In the first stage of our project, we will change the name of our module, the baud rate setting, and the password it will ask for connection. During configuration, there should be no connection to the module via Bluetooth. When there is no connection, the LED on the module will flash rapidly.

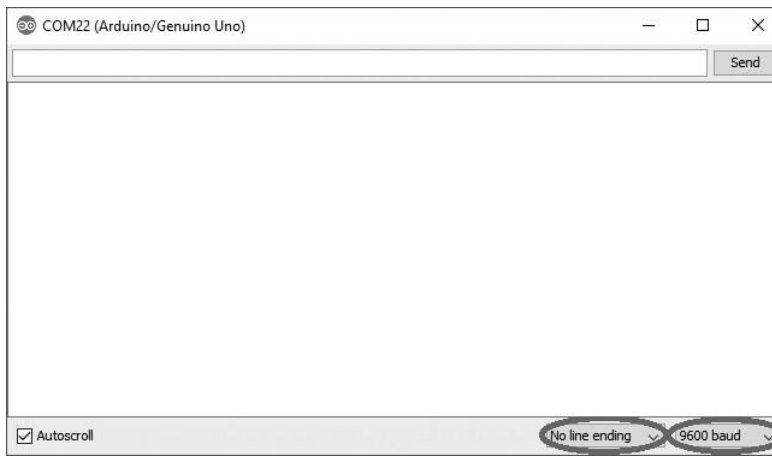
The default settings of the HC-06 module are as follows:

Name: linvor
Password: 1234
Baud rate: 9600

To change the settings, first, we upload the following code to our Tinylab:

```
void setup()  {
Serial.begin(9600);
  Serial1.begin(9600);    }    void
loop()         {                  if
(Serial1.available()) {         int
inByte      =      Serial1.read();
Serial.write(inByte);
   }    if (Serial.available())
{      int inByte =
Serial.read();
Serial1.write(inByte);
  }
}
```

We open the serial monitor screen in the Arduino IDE and make sure that the settings are as follows:



To test whether communication with the module has started, we can type "AT". The module should respond with "OK". Then, we can use the following commands to change its settings:

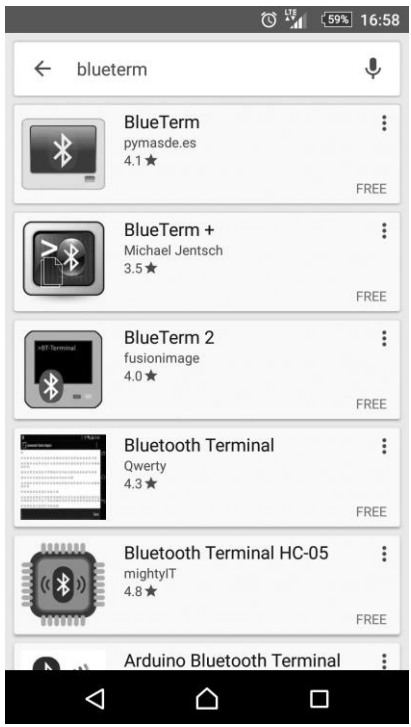
To change the module's name, use the command:

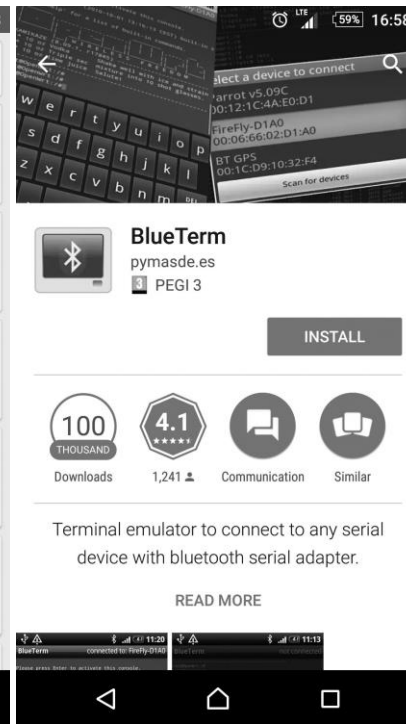
AT+NAMEkartismi

To change the password (PIN), use the command: **`AT+PIN1234`**

To change the baud rate, use the command: **`AT+BAUD4`** (1:1200, 2:2400, 3:4800, 4:9600, 5:19200, 6:38400, 7:57600, 8:115200, A:460800, B:921600, C:1382400).

After configuring the settings, we install the BlueTerm application on our Android device from Google Play:





After installing the application, we upload the following code to our Tinylab:

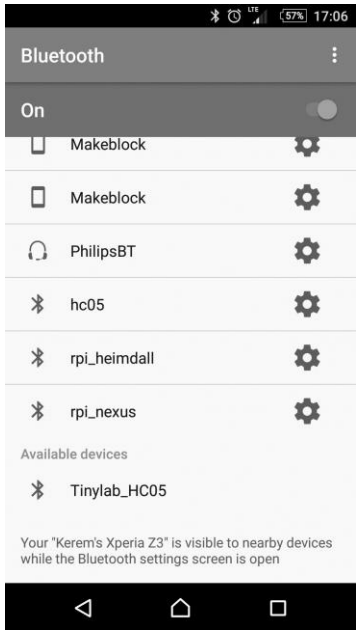
```
//LED pinleri
 #define led1 13
```

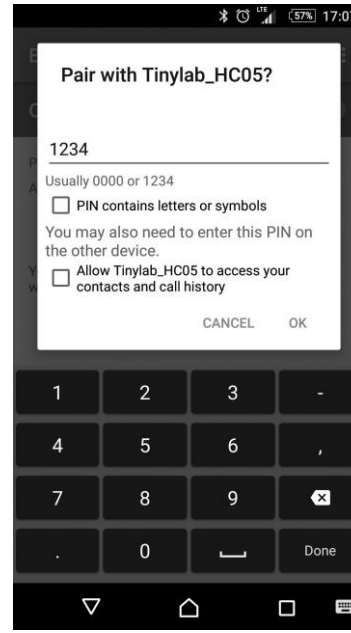
```
#define led2 12
#define led3 11
#define led4 10

void setup() {   //LED pinlerini cikis olarak
ayarla   pinMode(led1, OUTPUT);   pinMode(led2,
OUTPUT);   pinMode(led3, OUTPUT);
pinMode(led4, OUTPUT);   //HC06 ile seri
iletisimi 9600 baud'da baslat
  Serial1.begin(9600);
  Serial1.print("1 ile 4 arasinda bir LED numarasi girin veya
");
  Serial1.println("x ile hepsini sondurun");
}
void loop() {   if (Serial1.available()) //Haberlesme
baslayana kadar bekle   {     char ch = Serial1.read();
//Bluetooth uzerinden gelen karakteri oku     switch (ch)
//karaktere gore LED'leri yak     {       case '1':
       digitalWrite(led1, HIGH);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
digitalWrite(led4, LOW);
        Serial1.println("1 numarali LED yandi");
        break;       case
'2':
        digitalWrite(led1, LOW);         digitalWrite(led2,
HIGH);         digitalWrite(led3, LOW);
digitalWrite(led4, LOW);         Serial1.println("2
numarali LED yandi");         break;       case '3':
        digitalWrite(led1, LOW);         digitalWrite(led2,
LOW);         digitalWrite(led3, HIGH);
digitalWrite(led4, LOW);         Serial1.println("3
numarali LED yandi");         break;       case '4':
        digitalWrite(led1, LOW);         digitalWrite(led2,
LOW);         digitalWrite(led3, LOW);
```

```
digitalWrite(led4, HIGH);          Serial1.println("4
numarali LED yandi");          break;       case 'x':
        digitalWrite(led1, LOW);          digitalWrite(led2,
LOW);          digitalWrite(led3, LOW);
digitalWrite(led4, LOW);          Serial1.println("Tum LED'ler
sondu");          break;
      default:    //hatali    karakter    girisinde    uyari    yap
Serial1.println("Yanlis giris yapildi.");          break;     }
  }
}
```

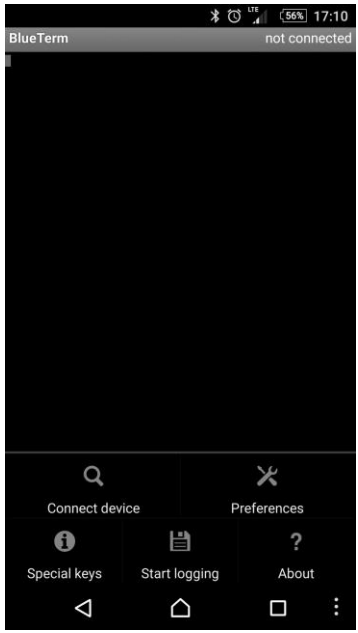
As keen readers will notice, this code is almost identical to the one in our 3rd serial communication lesson. The only difference is that we use Serial1 instead of Serial. This is because Tinylab's core, the Arduino Leonardo, uses the virtual serial port Serial for communication via USB, while it uses the physical serial port Serial1 for communication with external components like XBee and Bluetooth modules.

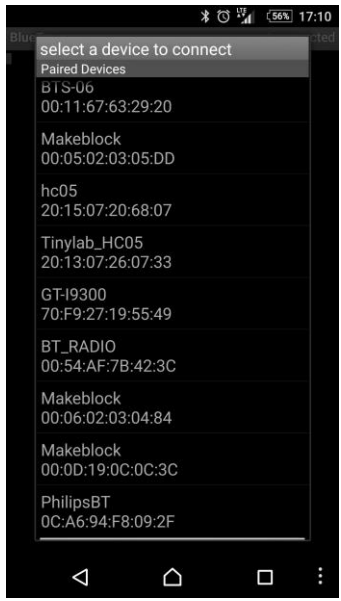
After uploading the code, we need to turn on the Bluetooth connection on our smart device and pair it with our HC06 module. We make sure our device discovers the module from the Bluetooth menu, and then complete the pairing process by entering the PIN code we set earlier:



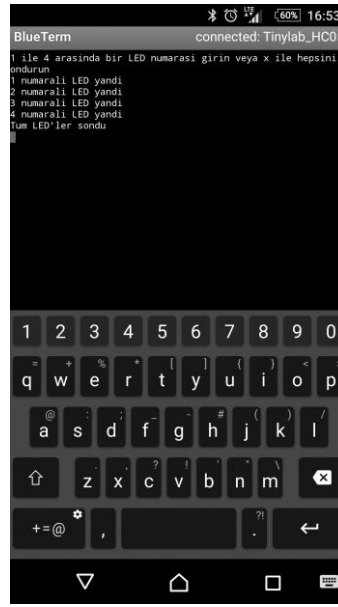


After completing the pairing process, we launch the BlueTerm application. Clicking on the "Connect device" option, we select the name we gave to our HC06 module from the list (the name I preferred was Tinylab_HC06).



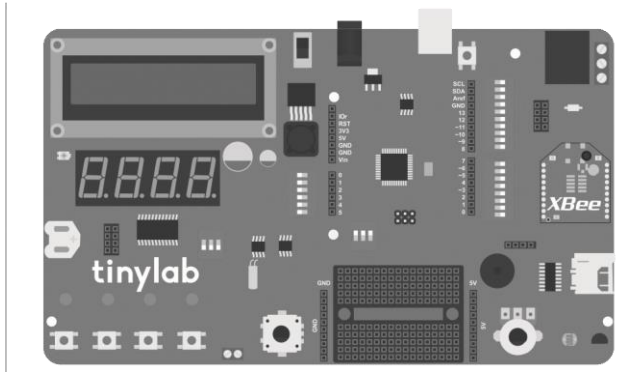


Once the connection is established, our project is ready to run.

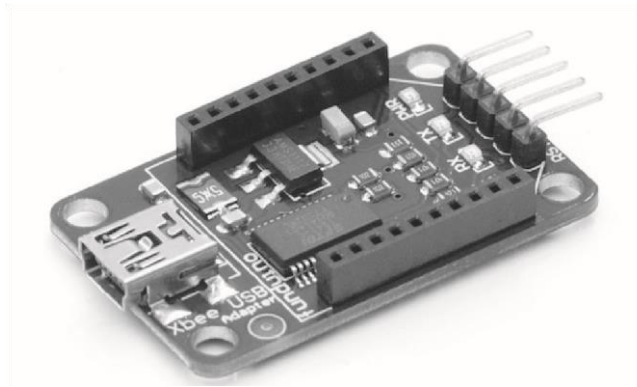


XBee Remote Sensor Module

XBee modules are commonly preferred wireless communication solutions in Arduino projects. XBee is the name given to the wireless communication devices produced by Digi. These modules use the IEEE 802.15.4 network protocol, providing point-to-point or multi-connection capabilities. The main difference from Zigbee, which is often confused with it, is that XBee is Digi's proprietary Zigbee protocol. There are many XBee modules on the market with different features. The simplest ones are the XBee 802.15.4 modules, also known as Series 1. When you obtain these modules, you can easily establish wireless communication between two devices without any configuration. In this project, we will use two devices: Tinylab and the XBee Explorer board.

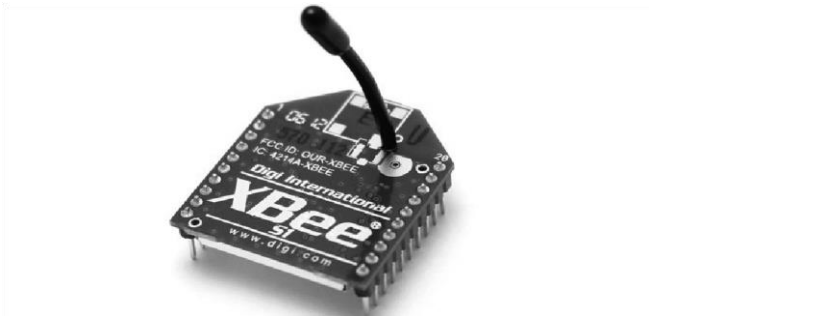


XBee modülü için bağlantı konektörü, Tinylab'imizde SD kart soketinin hemen üzerindedir.



Alt açıklama: XBee Explorer kartı

The XBee Explorer board is used to connect XBee modules to our computer via USB. Additionally, it allows us to change the settings of our XBee module and load different firmware to use different communication protocols using the XCTU program. XCTU is a comprehensive program; in this project, since we are using the Series 1 XBee module, we can easily communicate between our two devices without making any settings.



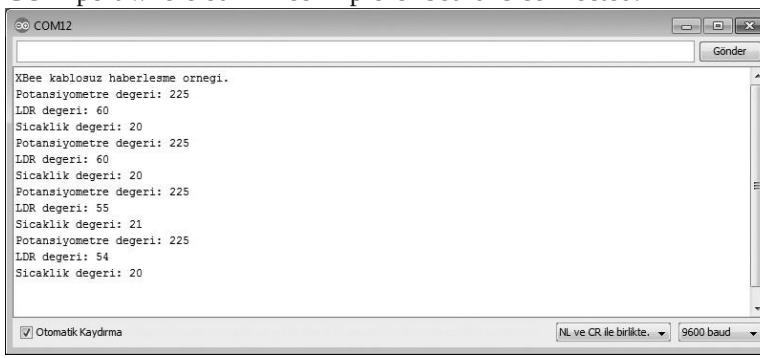
In this project, Tinylab will wirelessly send us the ambient temperature and humidity values via XBee.

The code:

```
//sensor pinleri
#define pot_pin A0
#define ldr_pin A2
#define lm35_pin A3

void setup()
{
  //seri iletisimi 9600 baud'da baslat
  Serial1.begin(9600);   //seri monitor acilana kadar
bekle   while (! Serial1);   Serial1.println("XBee
kablosuz haberlesme ornegi.");   delay(3000); }

void loop() {   //potansiyometreyi oku   int pot_val =
analogRead(pot_pin);   //ldr yi oku ve degeri lux e
cevir   int ldr_val = ((2500.0 / (analogRead(ldr_pin) *
(5.0 / 1024.0))) - 500) / 10.0;
  //lm35 i oku ve degeri santigrada cevir
  int lm35_val = (5.0 * analogRead(lm35_pin) * 100.0) / 1024;
Serial1.print("Potansiyometre degeri: ");
  Serial1.println(pot_val);
  Serial1.print("LDR degeri: ");
  Serial1.println(ldr_val);
```

```
  Serial1.print("Sicaklik degeri: ");
Serial1.println(lm35_val);
delay(500); }
```

We will observe the communication over the serial port with the XBee Explorer board and our XBee module connected to our computer. For this, we can use a serial terminal program like PuTTY, or we can use the serial port monitor of the Arduino IDE. All we need to do is select the COM port where our XBee Explorer board is connected.



MPU6050 6-axis Accelerometer and Gyroscope Board

MPU6050 is an IMU (Inertial Measurement Unit) sensor with a 3-axis gyroscope and a 3-axis accelerometer. It is used to measure the motion and acceleration of objects. IMUs like this one are fundamental sensors for unmanned aerial vehicles. They are also used in devices such as balance robots and camera stabilization tools.

The difference between a gyroscope and an accelerometer is that a gyroscope measures movement, while an accelerometer measures acceleration. As we know, the only acceleration acting on a stationary object is gravity.

Thus, by measuring the acceleration due to gravity, we can obtain the orientation of the object in three-dimensional space. The information obtained from the accelerometer will be in units such as m/s^2 or multiples of the force of gravity (g). A gyroscope, on the other hand, is used to measure the motion of an object around any axis. In a stationary position, a gyroscope will measure zero or a very small value. As the object moves, the gyroscope sensor will provide us with the speed of this movement in units like degrees per second.

The MPU6050 utilizes the I2C communication interface. The I2C communication interface, developed by Philips Semiconductors (now NXP), is a serial communication interface. It is often referred to by abbreviations such as I2C (Inter-Integrated Circuit), IIC, and TWI (Two-Wire Interface). It is very convenient to use because it uses only 2 communication lines (SCL: Serial Clock and SDA: Serial Data).

As we recall, the external EEPROM, real-time clock, and LCD display driver on Tinylab also use the I2C connection. Another advantage of the I2C connection is that devices on the same bus can be used simultaneously without issues as long as they have different addresses.

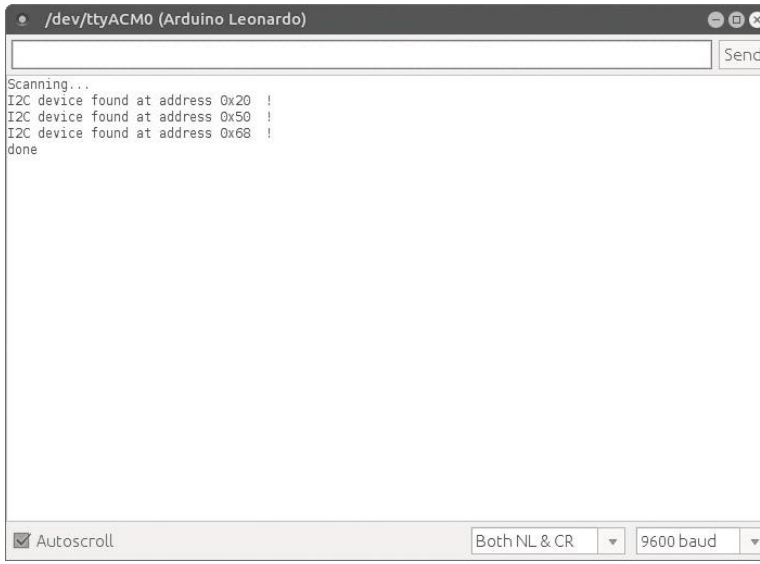
To see the addresses of the I2C devices on our Tinylab, we can use the following I2C scanner code:

```
// --------------------------------------
// i2c_scanner
//
// Version 1
//    This program (or code that looks like it) //
can be found in many places.
//    For example on the Arduino.cc forum.
//    The original author is not know.
// Version 2, Juni 2012, Using Arduino 1.0.1
//     Adapted to be as simple as possible by Arduino.cc user
Krodal
// Version 3, Feb 26  2013
```

```
//    V3 by louarnold
// Version 4, March 3, 2013, Using Arduino 1.0.3 //
by Arduino.cc user Krodal.
//    Changes by louarnold removed.
//    Scanning addresses changed from 0...127 to 1...119,
//    according to the i2c scanner by Nick Gammon
//    http://www.gammon.com.au/forum/?id=10896
// Version 5, March 28, 2013 //    As version 4,
but address scans now to 127.
//    A sensor seems to use address 120.
// Version 6, November 27, 2015.
//    Added waiting for the Leonardo serial communication.
//
//
// This sketch tests the standard 7-bit addresses // Devices
with higher bit address might not be seen properly.
//

#include <Wire.h>
void setup() {
  Wire.begin();

  Serial.begin(9600);   while (!Serial);             //
Leonardo: wait for serial monitor   Serial.println("\nI2C
Scanner"); }

void loop() {   byte error,
address;   int nDevices;
Serial.println("Scanning...");

  nDevices = 0;   for (address = 1; address <
127; address++ )   {
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if     //
a device did acknowledge to the address.
```

```
Wire.beginTransmission(address);     error =
Wire.endTransmission();

    if (error == 0)     {       Serial.print("I2C device
found at address 0x");       if (address < 16)
Serial.print("0");
      Serial.print(address, HEX);
Serial.println("  !");

      nDevices++;
    }     else if (error == 4)     {
Serial.print("Unknow error at address 0x");
if (address < 16)         Serial.print("0");
      Serial.println(address, HEX);
   }
}
  if (nDevices == 0)     Serial.println("No
I2C devices found\n");   else
Serial.println("done\n");

  delay(5000);           // wait 5 seconds for next scan
}
```

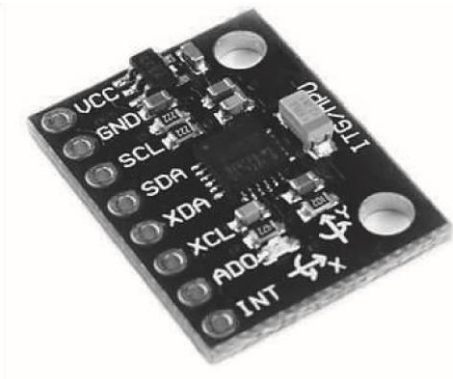
After uploading this code, when we open the serial port screen, we should encounter the following view:



The listed I2C devices here are as follows:

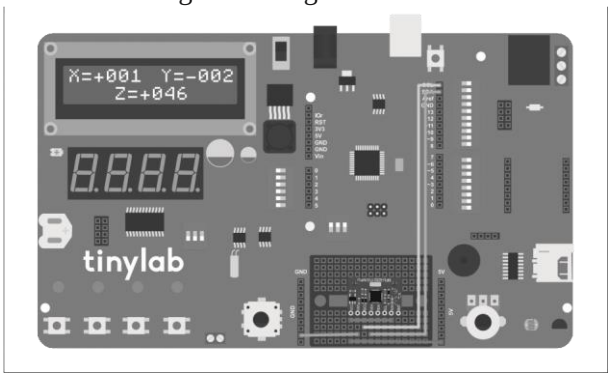
- 0x20 -> LCD screen driver (MCP23008 I/O expander)
- 0x50 -> External EEPROM (24LC256)
- 0x68 -> Real time clock (DS1307)

The MPU6050 sensor board we will be using looks like this:

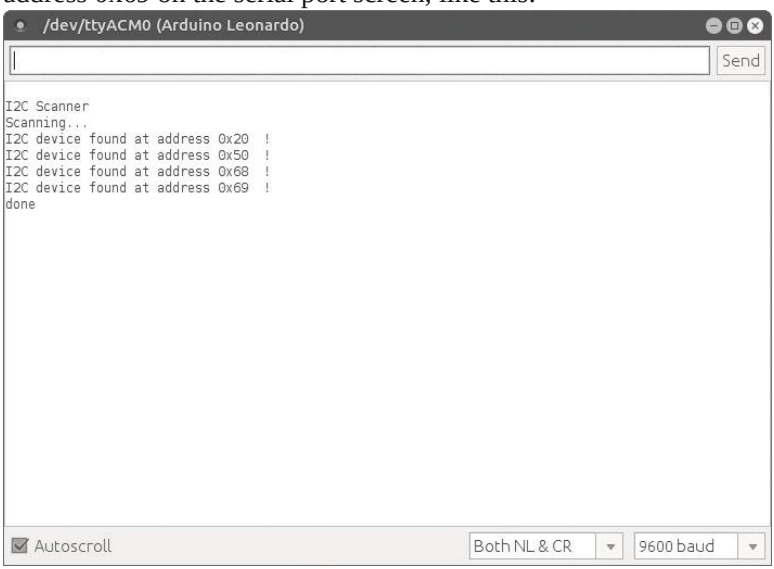


The sensor's X, Y, and Z axes are indicated on the module, and its default I2C address is 0x68. Remember, this address is also used by the

DS1307 integrated circuit on our Tinylab. To avoid address conflicts like this, we can use the AD0 pin on the module. When we provide 5V to this pin, the address of our MPU6050 will change to 0x69. We connect our sensor according to the diagram below and rerun our I2C scanner code:



When we run the code, we should see an additional I2C device with address 0x69 on the serial port screen, like this:



If we see the device with address 0x69 in the list, it means our connections are correct. Now, we can upload the code we will use with our sensor:

```
//gerekli kutuphaneler: I2C icin Wire.h,
//I2C LCD icin LiquidTWI2.h
#include <Wire.h>
#include <LiquidTWI2.h>

//MPU6050 sensorun adresi
#define MPU_addr 0x69

//I2C LCD icin tanimlama
LiquidTWI2 lcd(0);

//baslangic degerleri
int AcX = 0;
int AcY = 0; int
AcZ = 0;

void setup() {   //I2C LCD icin ayar
lcd.setMCPType(LTI_TYPE_MCP23008);
//I2C LCD icin ayar   lcd.begin(16,
2);   //arka isigi ac
lcd.setBacklight(HIGH);   //I2C
iletisimi baslat
  Wire.begin();
  //I2C haberlesme adresini MPU6050'nin adresi olarak belirle
  Wire.beginTransmission(MPU_addr);   //MPU6050'nin
PWR_MGMT_1 register'ina (0x6B adresinde) 0 gondererek
sensorun uyanmasini sagliyoruz   Wire.write(0x6B);
 Wire.write(0);
  //haberlesmeyi bitir
  Wire.endTransmission(true); }

void loop() {   //sensoru oku   readIMU(MPU_addr);
//sensor bilgisini ekrana yazdir   printToLCD();   //333ms
bekle, sensorun guncellenmesi icin optimum sure
delay(333); }
```

```
//sensoru okuma fonksiyonu void
readIMU(int addr)
{
  //I2C haberlesme adresini MPU6050'nin adresi olarak belirle
  Wire.beginTransmission(addr);
  //ACCEL_XOUT_H registerini (0x3B adresinde) oku
  Wire.write(0x3B);
  //haberlesmeye devam et
  Wire.endTransmission(false);
  //toplam 6 adet register'dan bilgi oku
  Wire.requestFrom(MPU_addr, 6, true);   //her eksenin bilgisi
toplam 2 byte (16-bit) oldugundan iki ayri register'da
tutuluyor   AcX = Wire.read() << 8 | Wire.read(); // 0x3B
(ACCEL_XOUT_H) ve 0x3C (ACCEL_XOUT_L)   //16-bit veriyi -100
ile +100 arasinda olcekle
  AcX = map(AcX, -32768, 32767, -100, 100);   AcY =
Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) ve 0x3E
(ACCEL_YOUT_L)   //16-bit veriyi -100 ile +100 arasinda
olcekle
  AcY = map(AcY, -32768, 32767, -100, 100);   AcZ =
Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) ve 0x40
(ACCEL_ZOUT_L)   //16-bit veriyi -100 ile +100 arasinda
olcekle
  AcZ = map(AcZ, -32768, 32767, -100, 100);
}

//ivme verisini LCD ekrana yaz void
printToLCD() {   //imleci birinci satir
birinci sutuna getir   lcd.setCursor(0, 0);
lcd.print("X=");

  //x-ekseni degeri pozitif ise
if (AcX >= 0)
 {
    //"X=" den sonra "+" isareti koy     lcd.setCursor(2, 0);
lcd.print('+');     //AcX degiskenini basamak basamak LCD'ye
```

```
yaz     lcd.setCursor(3, 0);     lcd.print((AcX / 100) % 10);
lcd.setCursor(4, 0);     lcd.print((AcX / 10) % 10);
lcd.setCursor(5, 0);     lcd.print(AcX % 10);   }   //x-ekseni
degeri negatif ise   if (AcX < 0)   {     //"X=" den sonra "-"
isareti koy     lcd.setCursor(2, 0);     lcd.print('-');
//AcX degiskenini basamak basamak LCD'ye yaz
lcd.setCursor(3, 0);     lcd.print(abs((AcX / 100) % 10));
lcd.setCursor(4, 0);     lcd.print(abs((AcX / 10) % 10));
lcd.setCursor(5, 0);     lcd.print(abs(AcX % 10));   }

  //imleci birinci satir onbirinci sutuna getir
lcd.setCursor(10, 0);   lcd.print("Y=");   //y-ekseni degeri
pozitif ise   if (AcY >= 0)   {     //"Y=" den sonra "+"
isareti koy     lcd.setCursor(12, 0);     lcd.print('+');
    //AcY degiskenini basamak basamak LCD'ye yaz
lcd.setCursor(13, 0);     lcd.print((AcY / 100) % 10);
lcd.setCursor(14, 0);     lcd.print((AcY / 10) % 10);
lcd.setCursor(15, 0);     lcd.print(AcY % 10);   }   //y-ekseni
degeri negatif ise   if (AcY < 0)   {     //"Y=" den sonra "-"
isareti koy     lcd.setCursor(12, 0);     lcd.print('-');
lcd.setCursor(13, 0);     //AcY degiskenini basamak basamak
LCD'ye yaz     lcd.print(abs((AcY / 100) % 10));
lcd.setCursor(14, 0);     lcd.print(abs((AcY / 10) % 10));
lcd.setCursor(15, 0);     lcd.print(abs(AcY % 10));   }
//imleci ikinci satir altinci sutuna getir   lcd.setCursor(5,
1);   lcd.print("Z=");   //z-ekseni degeri pozitif ise   if
(AcZ >= 0)   {     //"Z=" den sonra "+" isareti koy
lcd.setCursor(7, 1);     lcd.print('+');     //AcZ degiskenini
basamak basamak LCD'ye yaz     lcd.setCursor(8, 1);
lcd.print((AcZ / 100) % 10);     lcd.setCursor(9, 1);
    lcd.print((AcZ / 10) % 10);
lcd.setCursor(10, 1);     lcd.print(AcZ % 10);
}   //z-ekseni degeri negatif ise   if (AcZ < 0)
{     //"Z=" den sonra "-" isareti koy
lcd.setCursor(7, 1);     lcd.print('-');
//AcZ degiskenini basamak basamak LCD'ye yaz
```

```
lcd.setCursor(8, 1);     lcd.print(abs((AcZ /
100) % 10));     lcd.setCursor(9, 1);
lcd.print(abs((AcZ / 10) % 10));
lcd.setCursor(10, 1);     lcd.print(abs(AcZ %
10));   }
}
```

With this code, we display the sensor data we obtained from the MPU6050 accelerometer on the LCD screen of Tinylab. Since the sensor's output is 16-bit, the measured value for each axis will vary between -32768 and +32767. For practical use, we scale these values to a range of -100 to +100 using a simple map command.

Since the default sensitivity of the accelerometer is ±2g, when we place Tinylab on a flat surface, we should see a value of approximately +50 for the Z-axis. This indicates that an acceleration of approximately 1g, or the magnitude of gravitational force, is being measured along the Z-axis. Similarly, when we hold Tinylab on different edges, the gravitational force affects that axis, so the axis value will be shown as + or - 50.

Wireless Gamepad with nRF24L01

In this project, we are going to create a wireless game controller using the USB HID feature of the ATmega32u4 microcontroller at the heart of Tinylab.

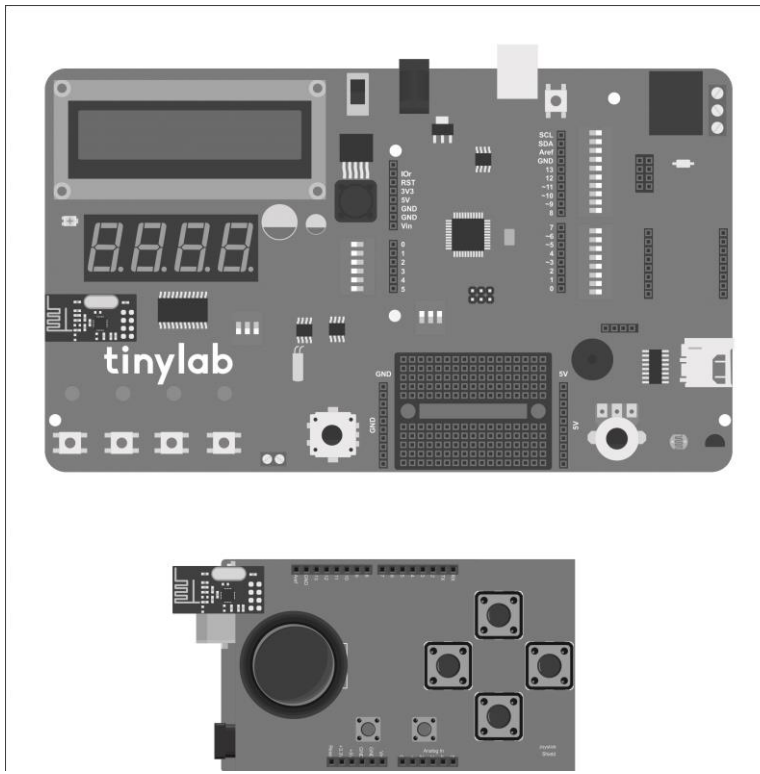
In this project, we are going to communicate between 2 Arduinos using nRF24L01 modules. One Arduino has a joystick shield, and we use this Arduino as the gamepad. The values of the analog joystick and buttons on the shield are read and sent to Tinylab, which is connected to our computer. Since Tinylab identifies itself as a gamepad to the computer, the received sensor data is used to report joystick and button movements to our computer.

The Material List:

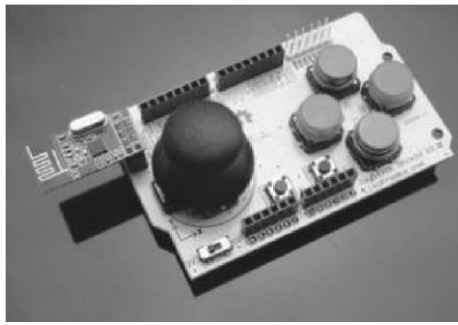
- Tinylab
- 1 x Arduino Uno
- 2 x nRF24L01

- 1 x Arduino Joystick Shield

The nRF24L01 wireless communication module is a commonly used wireless communication solution in Arduino projects. It operates at 2.4GHz frequency and connects via the SPI interface. In our project, we will use one of these modules as the receiver on Tinylab, and the one connected to the Arduino Uno with Joystick Shield will be set as the transmitter.



The Arduino Joystick Shield we will use in the project comes with a ready socket for the nRF24L01 wireless module. This makes it very convenient to use.



We will create a gamepad by attaching the joystick shield to our Arduino Uno. The connections for the Joystick Shield are as follows:

Analog X axis -> A0
Analog Y axis -> A1
Button A ->D2
Button B -> D3
Button C -> D4
Button D -> D5
Button E -> D6
Button F -> D7
Joystick Buton -> D8

The code we will upload retrieves the values of these buttons and axes, storing them in a variable of type String. Later, this String is sent via the nRF transmitter.

```
//Gerekli kutuphaneler: SPI haberlesme icin SPI.h,
//nRF24L01 modulu icin nRF24L01p.h
#include <SPI.h>
#include <nRF24L01p.h>

//nRF24L01 modulunun joystick shield uzerindeki CE ve CSN pin
baglantilari #define CE_PIN 9
#define CSN_PIN 10
```

```
//joystick shield uzerindeki butonlar ve eksenler
#define x_axis A0 // x ekseni
#define y_axis A1 //y y ekseni
#define button1 8 // joystick butonu
#define button2 2 // A butonu
#define button3 3 // B butonu
#define button4 4 // C butonu
#define button5 5 // D butonu
#define button6 6 // E butonu #define
button7 7 // F butonu

nRF24L01p transmitter(CSN_PIN, CE_PIN); //CSN,CE

//haberlesmede kullanilacak veriyi String tipinde bir
degiskende tutuyoruz String dataString = "";

void setup() {   delay(150);
//SPI haberlesmeyi baslat
  SPI.begin();   //SPI haberlesmede bit sirasini en anlamli
bit (MSB) ilk gelecek sekilde ayarla
SPI.setBitOrder(MSBFIRST);   //nRF modulunun haberlesme
kanalini 90 olarak ayarla   transmitter.channel(90);
//verici rolundeki nRF modulunun adresini tiny olarak
belirliyoruz   transmitter.TXaddress("tiny");   //vericiyi
baslat   transmitter.init(); }

void loop() {   //baslangic karakteri   dataString +=
"$";   //x ekseni degeri 1000'den kucuk ise bir 0 ekle
if(analogRead(x_axis) < 1000) dataString += "0";   //x
ekseni degeri 100'den kucuk ise bir 0 ekle
if(analogRead(x_axis) < 100) dataString += "0";   //x
ekseni degeri 10'dan kucuk ise bir 0 ekle
if(analogRead(x_axis) < 10) dataString += "0";   //x
ekseni degerini gonderilecek String'e ekle
dataString += String(analogRead(x_axis));   dataString
+= ",";   //y ekseni degeri 1000'den kucuk ise bir 0
ekle   if(analogRead(y_axis) < 1000) dataString +=
```

```
"0";   //y ekseni degeri 100'den kucuk ise bir 0 ekle
if(analogRead(y_axis) < 100) dataString += "0";   //y
ekseni degeri 10'dan kucuk ise bir 0 ekle
if(analogRead(y_axis) < 10) dataString += "0";   //y
ekseni degerini gonderilecek String'e ekle
dataString += String(analogRead(y_axis));   dataString
+= ",";   //buton1 degerini gonderilecek String'e ekle
dataString += String(digitalRead(button1));
dataString += ",";   //buton2 degerini gonderilecek
String'e ekle   dataString +=
String(digitalRead(button2));   dataString += ",";
//buton3 degerini gonderilecek String'e ekle
dataString += String(digitalRead(button3));
dataString += ",";   //buton4 degerini gonderilecek
String'e ekle   dataString +=
String(digitalRead(button4));   dataString += ",";
//buton5 degerini gonderilecek String'e ekle
dataString += String(digitalRead(button5));
dataString += ",";   //buton6 degerini gonderilecek
String'e ekle   dataString +=
String(digitalRead(button6));   dataString += ",";
//buton7 degerini gonderilecek String'e ekle
dataString += String(digitalRead(button7));
dataString += "#";    //String'i gonder
  transmitter.txPL(dataString);   //gonderme metodu
hizli, karsi taraftan alindi mesajini bekleme
transmitter.send(FAST);   //String'i bosalt   dataString
= "";   delay(50); }
```

On the receiver side, the incoming String is split into its components, and the joystick axes and buttons are moved accordingly based on these values.

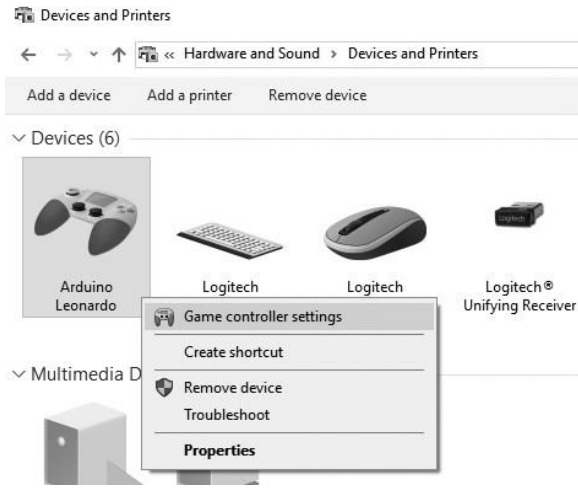
```
//Gerekli kutuphaneler: SPI haberlesme icin SPI.h,
//nRF24L01 modulu icin nRF24L01p.h,
//Bilgisayarin Tinylab'i joystick olarak taniyabilmesi icin
Joystick.h
```

```
#include <SPI.h>
#include <nRF24L01p.h>
#include <Joystick.h>

//nRF24L01 modulunun Tinylab uzerindeki CE ve CSN pin
baglantilari #define CE_PIN 9
#define CSN_PIN 8

//Joystick kullaninimi icin tanimlama
Joystick_ Joystick;

//varsayilan degerler
int pos_x = 0; int
pos_y = 0; int a_btn
= 0; int b_btn = 0;
int c_btn = 0; int
d_btn = 0;
int e_btn = 0; int
f_btn = 0; int
joy_btn = 0;

//haberlesmede kullanilacak veriyi String tipinde bir
degiskende tutuyoruz String message;

nRF24L01p receiver(CSN_PIN, CE_PIN); //CSN,CE

void setup() {   delay(150);
//SPI haberlesmeyi baslat
  SPI.begin();   //SPI haberlesmede bit sirasini en anlamli
bit (MSB) ilk gelecek sekilde ayarla
SPI.setBitOrder(MSBFIRST);
  //USB joystick olarak tanit
  Joystick.begin();   //nRF modulunun haberlesme kanalini
90 olarak ayarla   receiver.channel(90);   //alici
rolundeki nRF modulunun adresini tiny olarak belirliyoruz
```

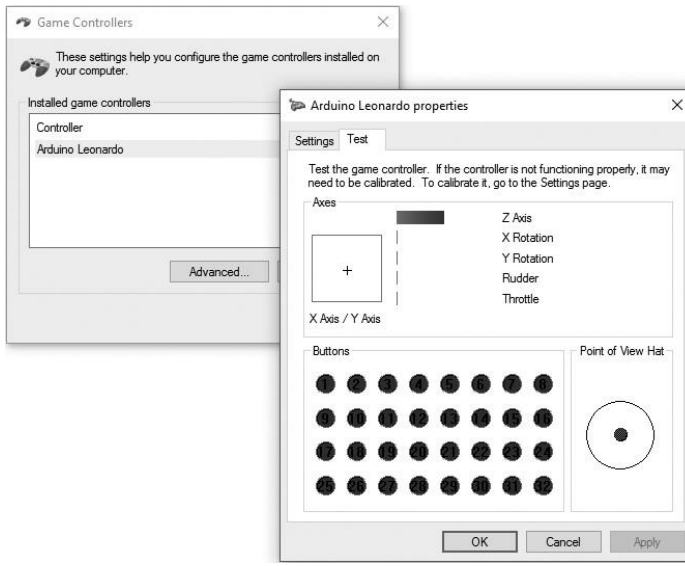
```
receiver.RXaddress("tiny");   //aliciyi baslat
receiver.init(); }

void loop() {   //nRF modul uzerinden gelen degerleri oku
readValues();   //x ekseni degerini joystick olarak
bilgisayara bildir
  Joystick.setXAxis(pos_x);
  //y ekseni degerini joystick olarak bilgisayara bildir
  Joystick.setYAxis(pos_y);   //joystick shield uzerindeki
butonlar ters calistigi icin hepsinin tersini aliyoruz
Joystick.setButton(0, !a_btn);
  Joystick.setButton(1, !b_btn);
  Joystick.setButton(2, !c_btn);
  Joystick.setButton(3, !d_btn);
  Joystick.setButton(4, !e_btn);
  Joystick.setButton(5, !f_btn);
  Joystick.setButton(6, !joy_btn);
}

//nRF24L01 modulunden gelen degerleri degiskenlere tasiyan
fonksiyon void readValues() {   if
(receiver.available())//nRF modulu takili ise   {
//alici modulden gelen mesaji oku     receiver.read();
receiver.rxPL(message);     //gelen mesaji karakter
sirasina gore ayir, degiskenlere ata     //ilk karakter
"$", sonraki 4 karakter x ekseni degeri
    String joy_x = message.substring(1, 5);     //x ekseni
degerini int tipine cevir     pos_x = joy_x.toInt();     //x
ekseni degerinden sonra "," geliyor, sonraki 4 karakter y
ekseni degeri     String joy_y = message.substring(6, 10);
//y ekseni degerini int tipine cevir     pos_y =
joy_y.toInt();     //y ekseni degerinden sonra "," geliyor,
sonraki 1 karakter a butonu
    String btn_a = message.substring(11, 13);     //a butonu
degerini int tipine cevir     a_btn = btn_a.toInt();     //a
butonu degerinden sonra "," geliyor, sonraki 1 karakter b
```

```
butonu     String btn_b = message.substring(13, 15);     //b
butonu degerini int tipine cevir     b_btn = btn_b.toInt();
//b butonu degerinden sonra “,” geliyor, sonraki 1 karakter c
butonu     String btn_c = message.substring(15, 17);     //c
butonu degerini int tipine cevir     c_btn = btn_c.toInt();
//c butonu degerinden sonra “,” geliyor, sonraki 1 karakter d
butonu     String btn_d = message.substring(17, 19);     //d
butonu degerini int tipine cevir     d_btn = btn_d.toInt();
//d butonu degerinden sonra “,” geliyor, sonraki 1 karakter e
butonu     String btn_e = message.substring(19, 21);     //e
butonu degerini int tipine cevir     e_btn = btn_e.toInt();
//e butonu degerinden sonra “,” geliyor, sonraki 1 karakter f
butonu     String btn_f = message.substring(21, 23);     //e
butonu degerini int tipine cevir     f_btn = btn_f.toInt();
//e butonu degerinden sonra “,” geliyor, sonraki 1 karakter
joystick butonu     String btn_joy = message.substring(23,
25);     //joystick butonu degerini int tipine cevir
joy_btn = btn_joy.toInt();     //String’i bosalt     message =
“”;
  }
}
```

After uploading the code, if we have a Windows operating system installed on our computer, we open the Control Panel and click on "Devices and Printers". Here, we will see a device named "Arduino Leonardo" as a gamepad:

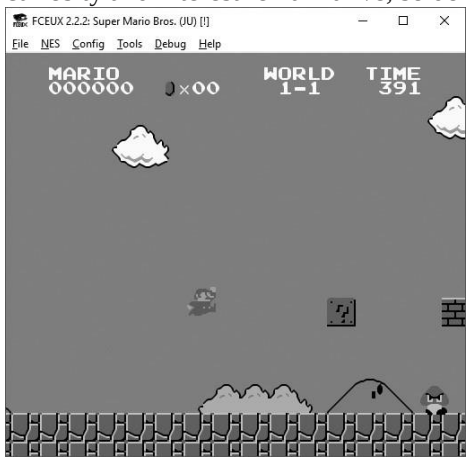


Right-clicking on the Arduino Leonardo and selecting "Game controller settings" allows us to check if our gamepad is functioning correctly.



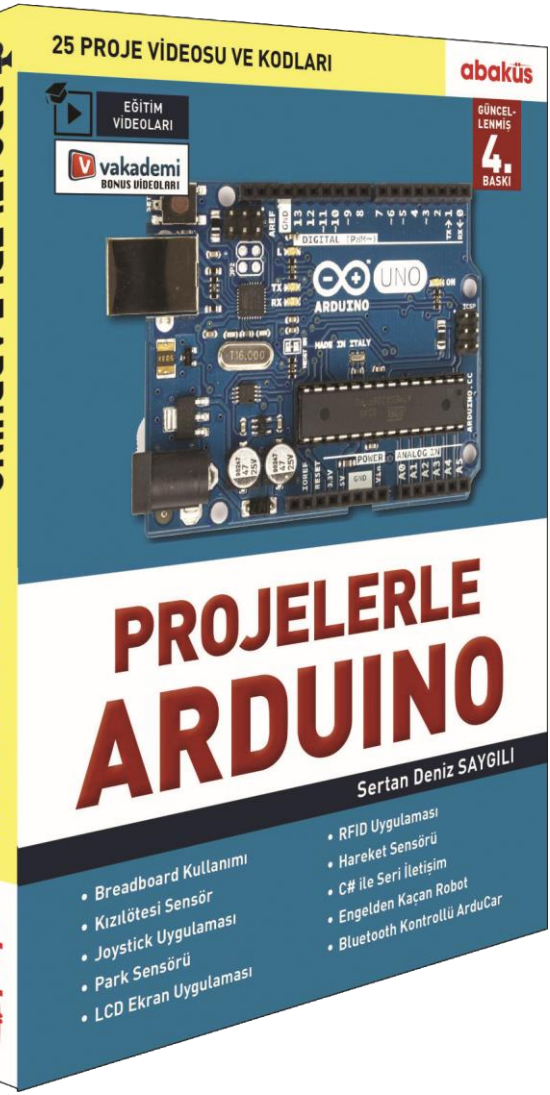
Our next step is to find a game to play. My choice is Super Mario :)

We've reached the final project in this book. Remember, what you can do with Arduino and Tinylab is not limited to what's in this book; the only limit is your imagination. I hope you've experienced the joy I felt while preparing this book as you read through it. The pleasure of creating, in my personal opinion, is an addictive feeling. I always hope your curiosity and interest remain alive, so don't be afraid to create!

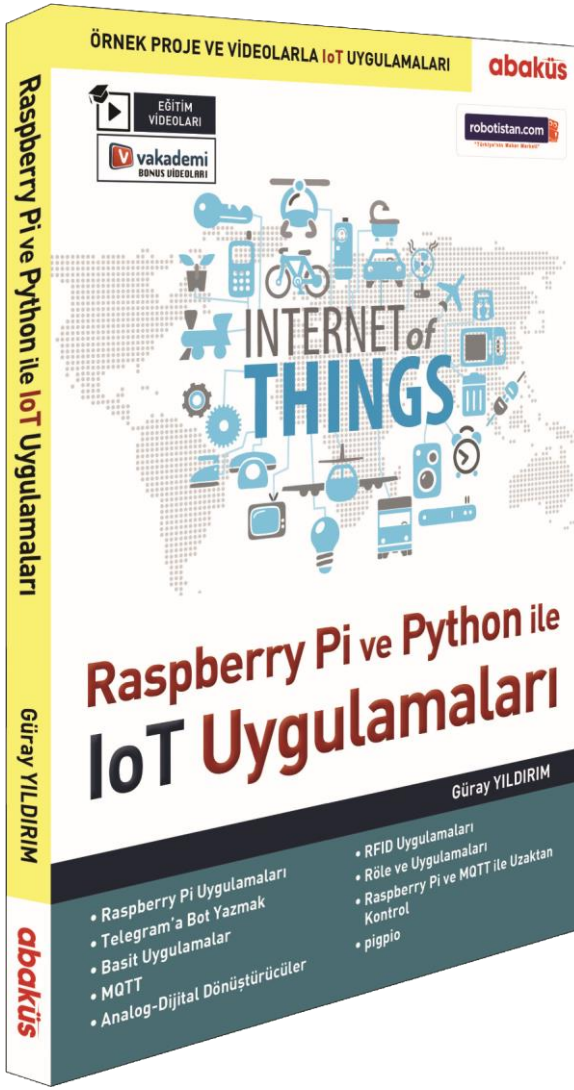


Source

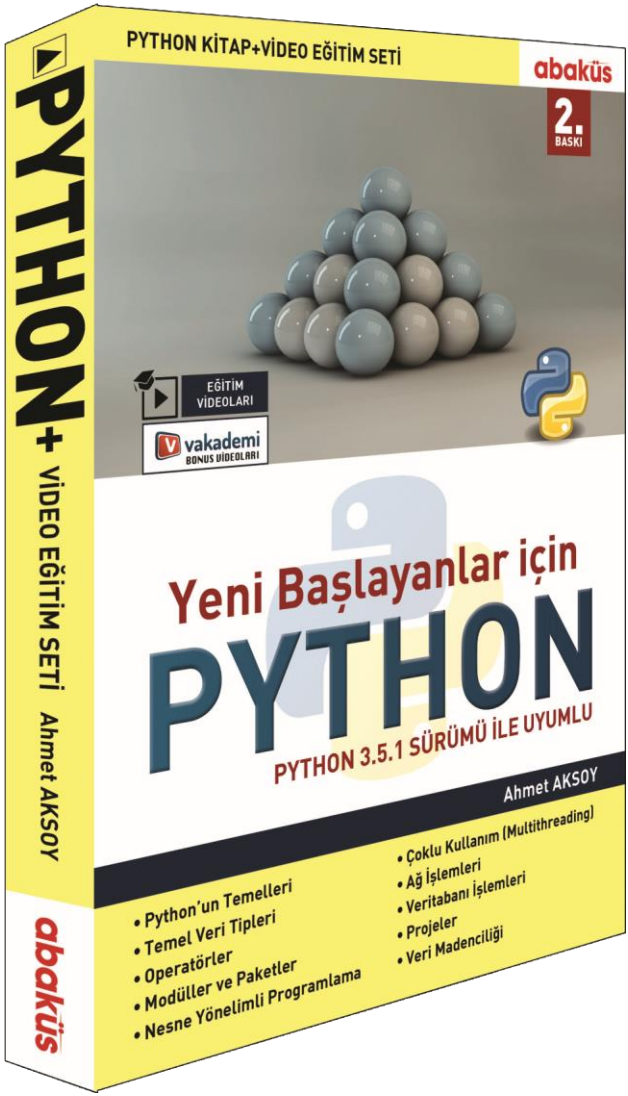
- Wikipedia: en.wikipedia.org
- Robotistan Maker Blog: maker.robotistan.com
- Arduino Playground: playground.arduino.cc
- Adafruit Blog: learn.adafruit.com
- Fritzing is used for thdrawings: fritzing.org



PROJELERLE ARDUINO Sertan Deniz SAYGILI



RASPBERRY PI VE PYTHON İLE IoT UYGULAMALARI Güray YILDIRIM



YENİ BAŞLAYANLAR İÇİN PYTHON Ahmet AKSOY