Electronics II
exercise 4

Deadline: Thu 19.2.2026 14:00

Wi-Fi and registers

About registers: Registers very small and fast bits of memory, directly accessible by processors (or drivers). All information in a computer goes through registers whenever it moves, with very few exceptions.

Working registers and state registers are two examples out of dozens of different register types. Working registers (työrekisteri) are used by the processor to perform operations, such as basic arithmetic. The processor can also use state registers (tilarekisteri) to, e.g., check interrupts and errors, or to control more complex functions of the device.

We'll take the port registers on an Arduino as an example. The microcontroller sets the states of a set of pins (port) by referring to registers corresponding to the port. Using functions of high-level languages, such as `digitalWrite()`, changes the values in these registers. The simple-looking function is in reality multiple device-specific machine instructions (konekäsky), which are not relevant to most users. These functions automate things such as accessing memory addresses, assigning registers and so on.

The same functionality as `digitalWrite()` can be achieved by placing correct values into certain registers. This allows performing multiple operations, such as flipping the state of multiple digital pins, in one command. Since the commands for altering the register values are closer to machine instructions, executing them is faster. A notable downside is that since the registers are hardware dependent, the code can't be used with other devices without modifications.

Read more here: Arduino - PortManipulation (arduino.cc)

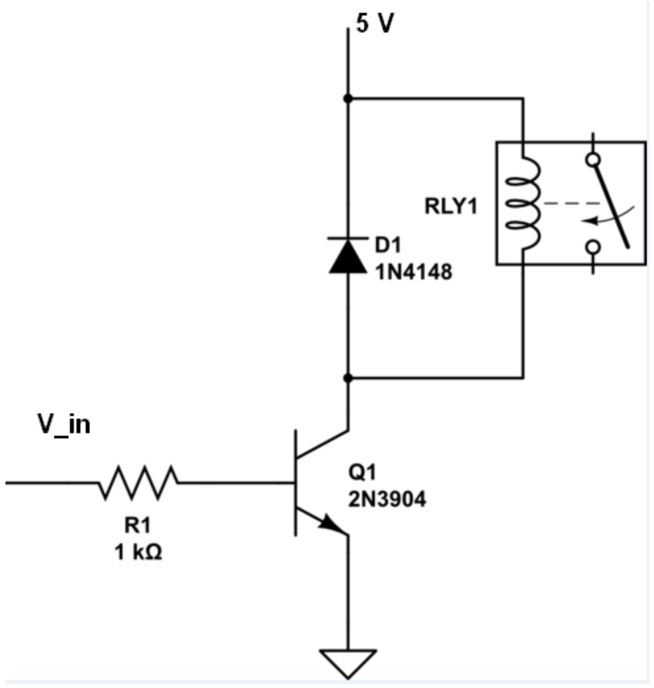
Register map of ATmega328P: ATmega328P datasheet, 30. Register Summary (microchip.com)

The course "Tietokoneen toiminta" discusses the use of registers and machine instructions in much more detail.

Task 1 – Clap switch (1p) [Arduino]

Use the microphone module to make a clap switch which toggles a relay when it hears two claps in short succession. You could use this to turn on any mains powered device, for example lights but since you're not an electrician you can settle for lighting up an LED or something.

NB: The relay can cause damage to the microcontroller if connected without a protective circuit. Use the circuit below.



Task 2 – Port manipulation (1p)

Read the four least significant bits of PORTD with internal pull-up resistors enabled, do a bitwise NOT on them and write them on the four most significant bits of the same port. Do not use `pinMode()`, `digitalRead()` or `digitalWrite()` functions in this task. Use port registers instead.

For testing connect the input pins to either GND or 5V (**3.3 V for ESP32**) and put leds on the output pins. Make sure the pull-ups don't get disabled when writing to the port. Also read the "Why use port manipulation?" section in the port manipulation article linked below.

Arduino - PortManipulation (arduino.cc), for ESP32 (instructables.com). This is the best beginner friendly guide for the ESP32 we found, even if it's in Portuguese. There are many ways to do this, including handling the raw register addresses, so feel free to look for yourself as well.

Task 3 – Temperature & humidity (1+1p) [Arduino]

a) Measure temperature and humidity with DHT11.

b) Measure the temperature of the microcontroller using the internal temperature sensor. Calibrate it to give approximately same readings with DHT11 (take the readings when microcontroller has been off for some minutes.). How much does the temperature rise when the microcontroller has been on for a while?

Task 4 – LED over Wi-Fi (2p) [ESP32]

Turn an LED on and read the position of a potentiometer over the Wi-Fi.

NB: This task may take some time. Make sure you start early and get in touch if run into difficulties.

Some ways to approach the task

Below are two ways to approach the task (and the final project). The assistants are familiar with these methods but there are many different ways to establish a wireless connection. You are encouraged to look for alternative methods if these two seem ill-suited for your needs.

Some companies offer cloud solutions, such as Arduino Cloud, which simplify IoT development significantly. While these solutions may be enticing, the free plans are often very restricted and the license terms may change unexpectedly. Cloud-solutions also require you to be connected to the Internet, which exposes the devices and data stream to attacks and interception. Self-hosting the IoT network takes some time and effort, but in return you have full control over every aspect of the system and can restrict it to a local network. As such, the usage of cloud services is not allowed on this course.

ESP32 and HTML

These tutorials give good tools to solve the task by making the ESP32 host a webpage. The ESP32 acts as an access point which you can connect to with a laptop/phone.

Pros: Ready UI, good for simple tasks, no need for a router/hotspot

Cons: Data processing on ESP32, HTML can be tedious to edit, UI can be inflexible

DFR0654 wiki - 10.3.1 WiFi Basic Tutorial (dfrobot.com)

Input Data on HTML Form ESP32/ESP8266 Web Server using Arduino IDE (randomnerdtutorials.com)

ESP32 and Python over TCP/IP connection

These tutorials give a good starting point on creating and managing ESP32s with Python. This is especially useful if you intend to handle large data sets and/or process the data in depth.

Pros: Data processing can be done on a laptop, capable of very fast data transfer, supports multiple microcontrollers

Cons: Requires a hotspot/router, programming can get involved, no ready-made UI

Search terms: "TCP/IP", "socket"

ESP32 Arduino Tutorial: Setting a socket server (dfrobot.com)

ESP32 Arduino Tutorial: Sending data with socket client (dfrobot.com)

Setup Communication between desktop PC and ESP32 over wifi (arduino.cc)

These may be useful in your final project

ESP32 Useful Wi-Fi Library Functions (Arduino IDE) (randomnerdtutorials.com)

How to use ESP8266/ESP32 as a TCP server and client with sockets (engineersgarage.com)