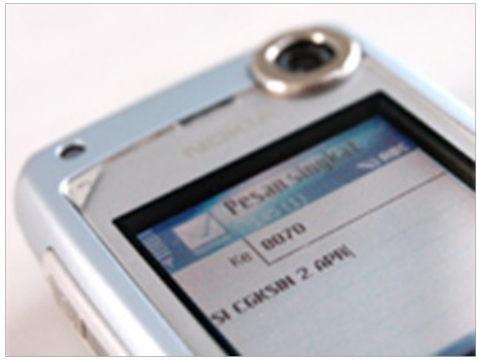


Computer Architectures

2. Implementing the door lock with Arduino

Prepared by: **Gábor Horváth**, ghorvath@hit.bme.hu
Presented by: **Gábor Lencse**, lencse@hit.bme.hu

2024. márc. 7.
Budapest

Outline

- Aim of the lecture:
  - To show how easy it is to work with the highly integrated microcontrollers available today
- Outline:
  - Introducing Arduino
    - The hardware
    - How to program
    - Sensors, peripherals
  - Implementing the door lock
    - The display
    - The RFID reader
    - The keypad
    - The whole hardware
    - The whole source code

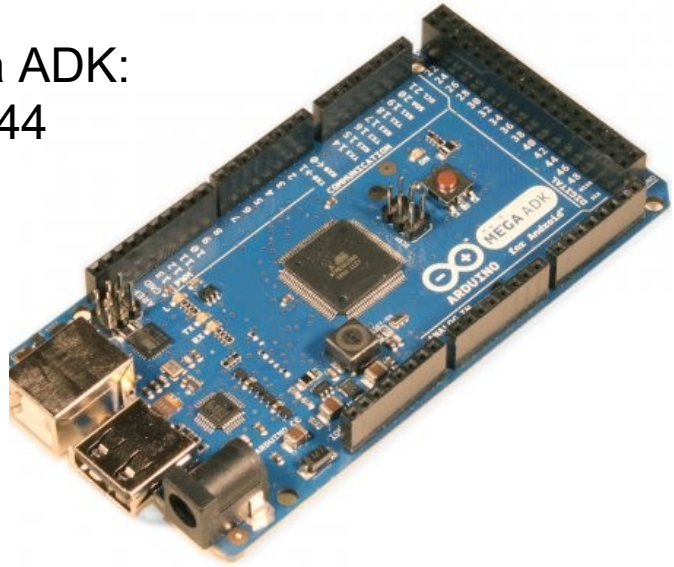
Arduino

- 2005, Ivrea, Italy
- Purpose: simple prototyping
- It is a family of prototyping boards (http://arduino.cc)
- Common features:
  - Atmel AVR 8 bit CPU (**Harvard** architecture!)
  - Integrated flash memory to store the program
  - Integrated RAM to be used as the main memory
  - Integrated EEPROM to be used as a non-volatile memory
  - Input/output capabilities:
    - Digital
    - Analog
  - It can be programmed through an USB port (not all of them)
  - C++-like language for software development

The Arduino Family

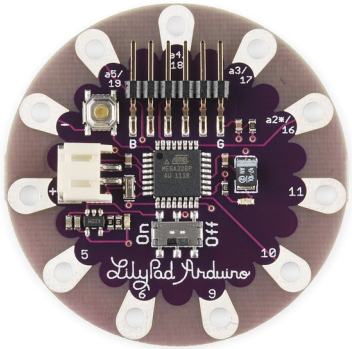
Uno:
€20

Mega ADK:
€44



Micro:
€18

LilyPad:
$25



The Arduino Family

- Differences:
  - Number of input/output pins
  - Size of the memory (flash/RAM/EEPROM)
  - Some special features:
    - The Mega ADK can be connected to Android devices
    - The LilyPad can be sewn to fabric

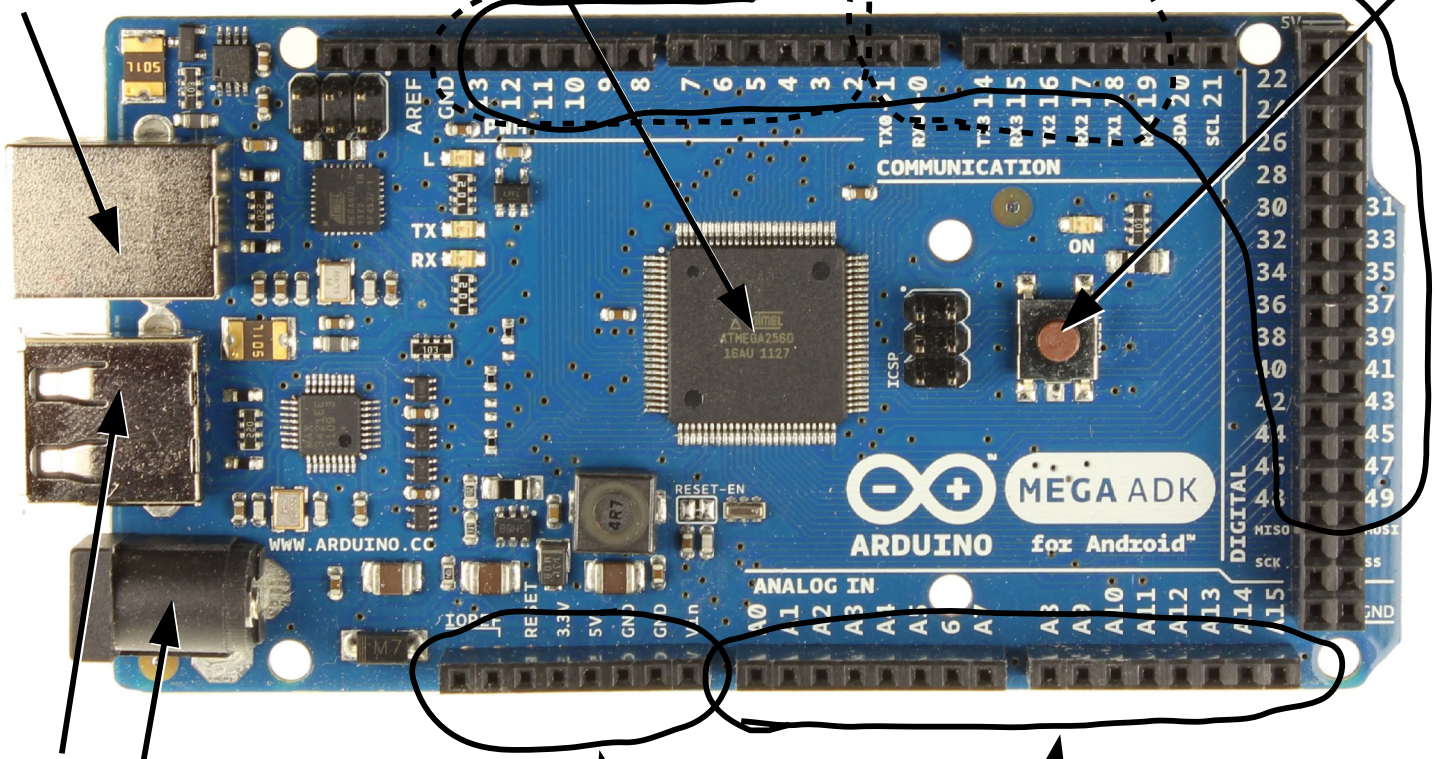
	CPU freq.	Flash	RAM	EEPROM	Digital I/O	Analog I/O
Uno	16 MHz	32 kB	2.5 kB	1 kB	14	6
Mega ADK	16 MHz	256 kB	8 kB	4 kB	54	16
Micro	16 MHz	32 kB	2,5 kB	1 kB	20	12
LilyPad	8 MHz	32 kB	2 kB	1 kB	9	4

Comparision

- Price
  - €44, for a 8 bit CPU, 16 MHz freq., and a couple kB or memory?
  - When there is the Raspberry Pi 3 for $35?

    (4x1200 MHz 64 bit ARM CPU, 1 GB RAM, strong GPU, HD movie playback, HDMI output, Ethernet port, Bluetooth, WiFi, etc.)
- They have different purposes:
  - Arduino:
    - Emphasis: I/O, all the time, as simply as possible
    - When switched on, it is up and running in 1 second
    - It is the largest model that costs €44, the cheap Leonardo is sufficient for most projects
  - Raspberry Pi:
    - Emphasis: general purpose computer (for teaching programming)
    - It requires an operating system (Linux)! Booting process takes a while.
    - It has input/output capabilities, but using them needs deep knowledge of the Linux kernel

Arduino Mega ADK

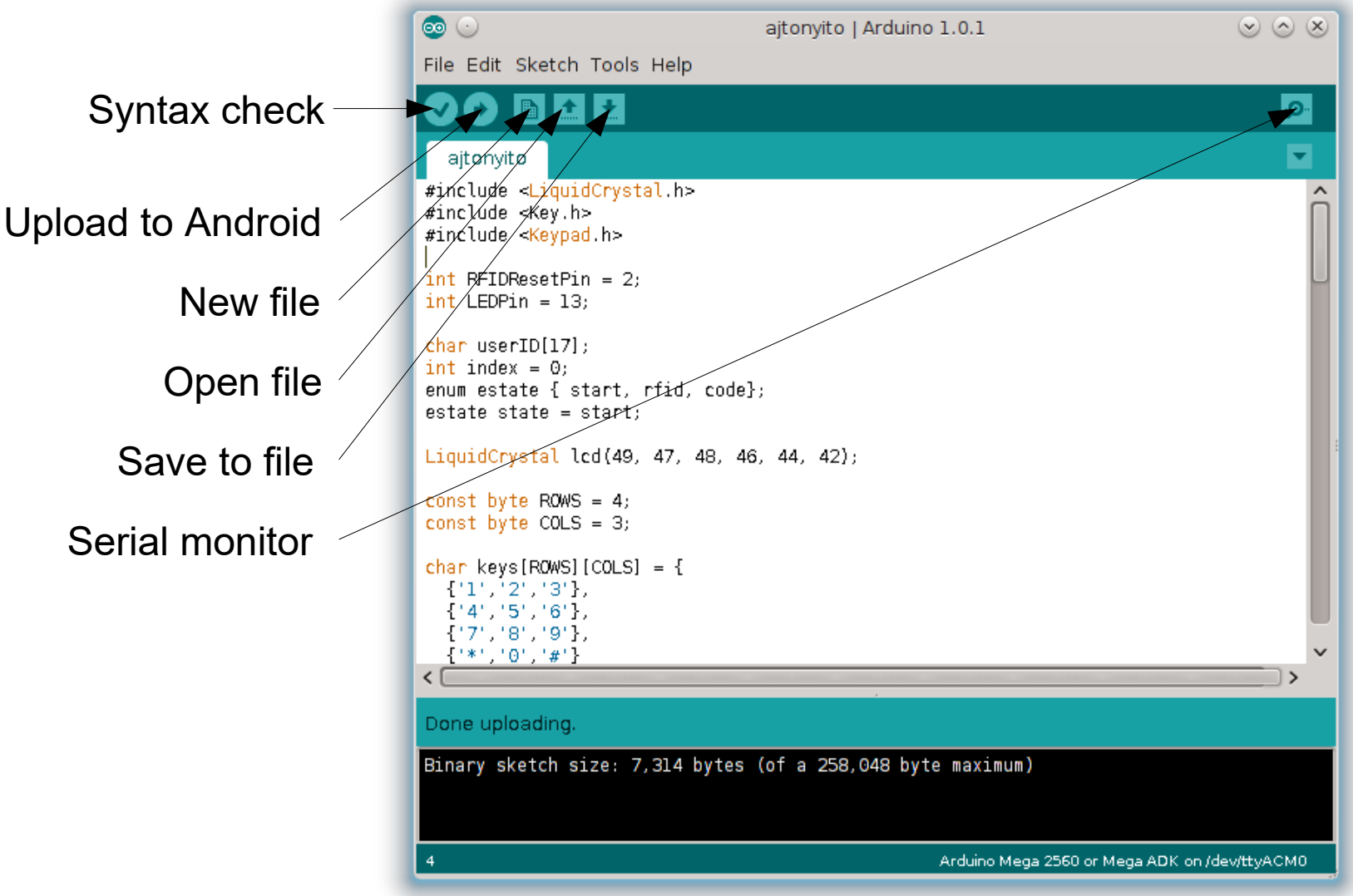


Programming

Programming

- Open-source cross-platform development environment (IDE)
  (http://arduino.cc/en/Main/Software)
- Language: C++-like (file extension: .ino)
- Compilation process:
  - IDE compiles AVR code (cross compiler)
  - It writes the program to the flash memory of the microcontroller through the USB port
- Debugging:
  - What the Arduino writes to its default serial port is transmitted to the PC through USB and displayed by the IDE

The development environment



The programming language

- Arduino calls the program a “sketch”
- Data types:
  - int: 16 bit integer
  - long: 32 bit integer
  - boolean: logical type (true/false), occupies 1 bit only in the memory (C++ has no such data type)
  - float: 32 bit floating point
  - char: stores ASCII characters (1 byte)
  - etc.
- Operators, loops, branches: like C++
- Class-es are allowed to use, and preprocessor directives as well
- There are two mandatory functions to write:
  - **void setup () { … }** - this function is executed once, when Arduino starts up
  - **void loop () { … }** - thus function runs after the initialization again and again (when it terminates, it is started again automatically)

Inputs/Outputs

Digital in- and outputs

- Digital pins can act as both input and output direction
  - But only one direction at a time
  - To set it up:
    - **pinMode (4, INPUT);** – pin 4 is set up to act as input
    - **pinMode (5, OUTPUT);** – pin 5 is set up to act as output
- Putting digital signals to digital pins:
  - **digitalWrite (5, HIGH);** – puts a logical 1 (5V) to pin 5
  - **digitalWrite (5, LOW);** – puts a logical 0 (0V) to pin 5
- Reading digital inputs:
  - **int val;**
  - **val=digitalRead (4);** – reads digital value from pin 4
  - **if (val==HIGH) …**, or **if (val==LOW) …**

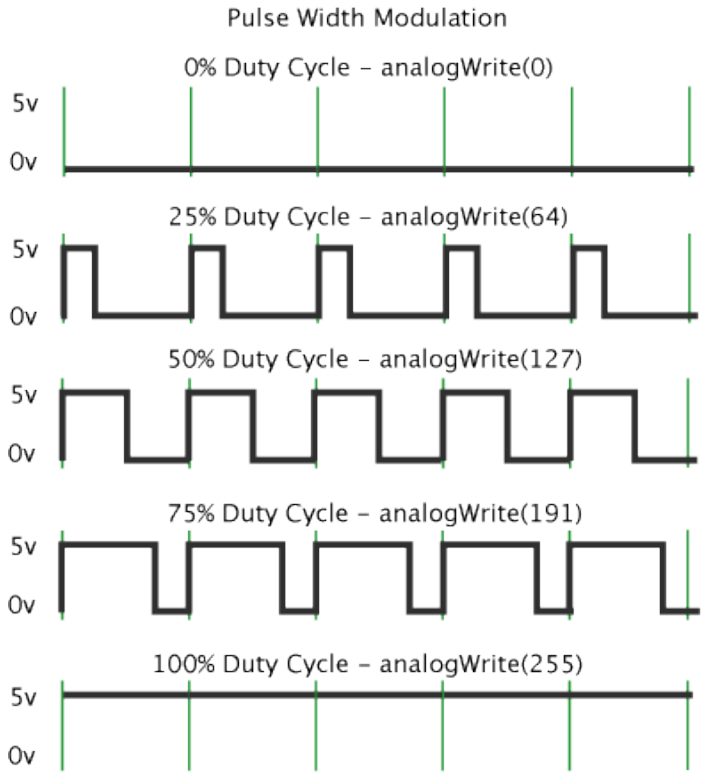
Analog inputs

- Does not detect only HIGH and LOW values
- ADC (Analog-Digital Converter) with 10 bit resolution
  → 1024 different voltage levels can be distinguished
- 1024 levels between 0V and 5V → 4.88 mV resolution
- Usage:
  - **int val;**
  - **val = analogRead (3);** - sample from analog input number 3
  - **val**: from 0 (in case of 0V) to 1023 (5V)
- The 5V maximum reference can be adjusted by calling **analogReference()**, but the maximum is 5V

PWM output

- It can not provide continuous analog output (e.g., 4.2V)
- But it can switch the output on and off very fast: it can produce any voltage in the “average” sense

  → **PWM: Pulse Width Modulation**
- This is done be calling function **analogWrite ()**
- Parameters: pin number, duty cycle (between 0…255)



Serial input and output

- Serial communication needs only 2 wires at most:
  - RX: for receiving bytes (if we need to receive bytes)
  - TX: for transmitting (sending) bytes (if needed)
- The Arduino ADK has 4 serial ports
  - The first one is used by the development environment for debugging purposes
- Usage: through **class Serial**
  - Pre-defined instances: **Serial**, **Serial1**, **Serial2**, **Serial3**
  - *Initialization*: **Serial1.begin (9600);** - open serial port 1 and set up speed to 9600 bps
  - *Writing*: **Serial1.write (…);** - sends a single byte, a NULL terminated string, or an array
  - *Writing*: **Serial1.print (...);** - the parameter is converted to string, and sends it. Function **Serial1.println (...);** adds an extra line break at the end as well.
  - *Reading*: **int received=Serial1.read();** - obtains a byte received (-1, if no bytes received)
  - *Check data availability*: **int count = Serial1.available();** - gives back the number of bytes received
  - *Close port*: **Serial1.end ();** - after closing the port, the pins can be used as general purpose in/out pins

Memory

Memory

- The microcontroller of Arduino has 3 kinds of memories
  - **Flash memory**: stores the program
    - Content is kept without power supply
  - **RAM**: stores the variables and the stack
    - Power supply is needed to keep content
  - **EEPROM**: to store non-volatile data
    - Content is kept without power supply
- AVR processors follow a Harvard architecture
  - There are 2 address spaces:
    - Instructions and constants are taken from the flash memory
    - For variables and stack, the RAM is used
  - And how to access the EEPROM?
    - It is treated as a peripheral (an I/O device)
    - There is a library to access it

EEPROM

- The EEPROM library is a standard component
- #include <EEPROM.h>
- Writing it:
  - **EEPROM.write (address, data);**
    - Address is int
    - Data is byte
- Reading it:
  - **byte a;**
    a = EEPROM.read (42);
    - Reads the 42th byte from the EEPROM

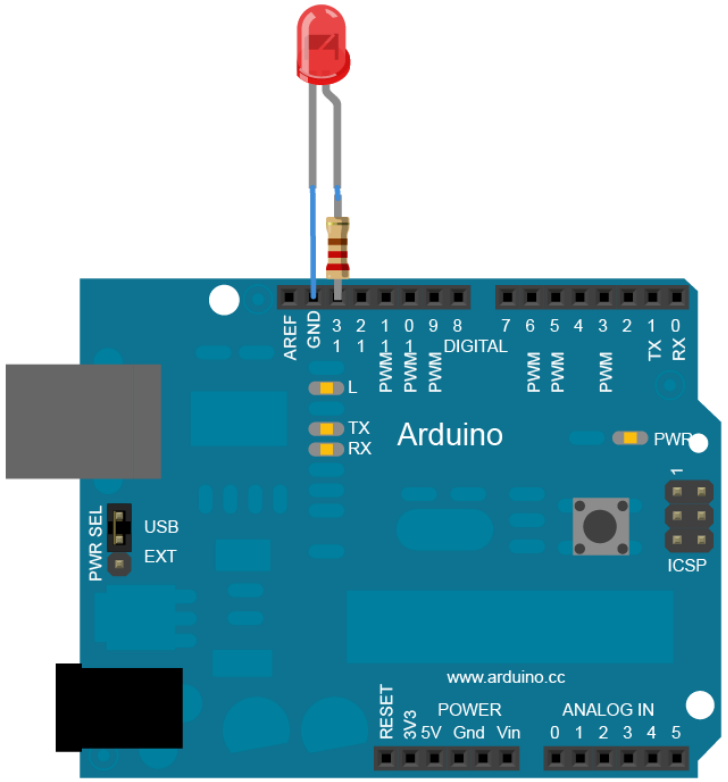
Connecting peripherals

Digital output peripherals

- **Example**:
  Blinking LED

  Components: LED, 220 Ohm resistor

  It even works without components!
  Pin 13 has a built-in LED.

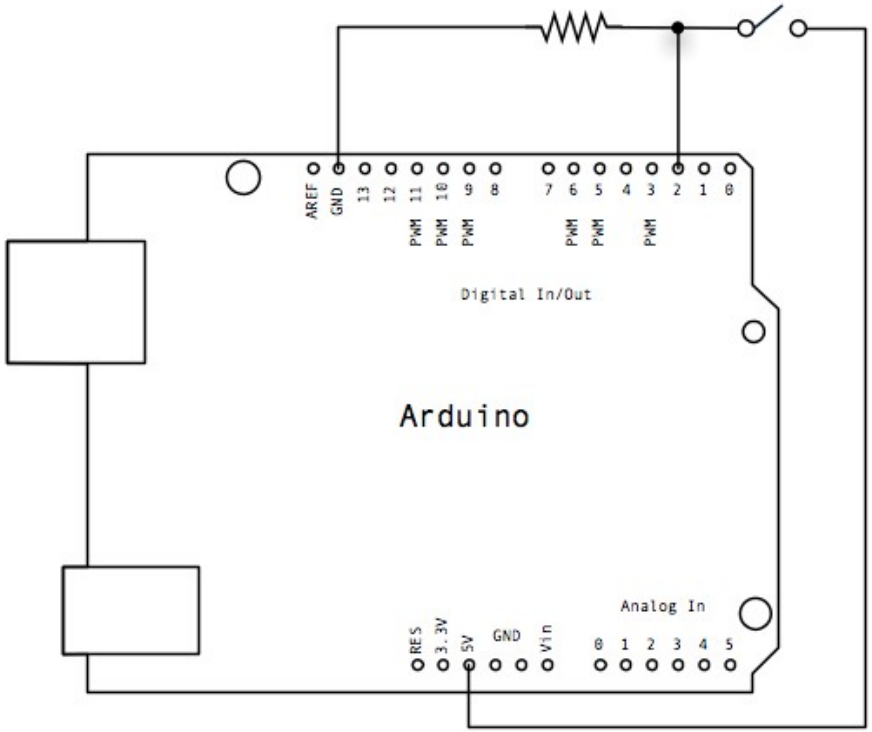


- **Code**:

```
const int ledPin = 13;
void setup () {
    pinMode (ledPin, OUTPUT);
}
void loop () {
    digitalWrite (ledPin, HIGH);
    delay (1000);
    digitalWrite (ledPin, LOW);
    delay (1000);
}
```

Digital input peripherals

- **Example**:
  Detecting the state of a button, and switching on the built-in LED accordingly

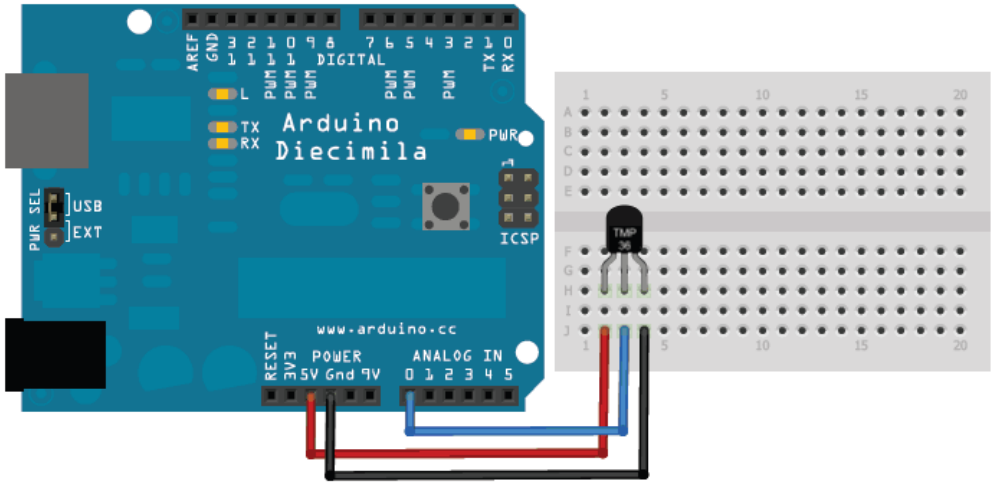


- **Code**:

```
const int buttonPin = 2;
const int ledPin =  13;
int buttonState = 0;
void setup () {
    pinMode (ledPin, OUTPUT);
    pinMode (buttonPin, INPUT);
}
void loop () {
    buttonState = digitalRead (buttonPin);
    digitalWrite (ledPin, buttonState);
}
```

Analog input peripherals

- **Example**:
  TMP36 temperature sensor



- **Code**:

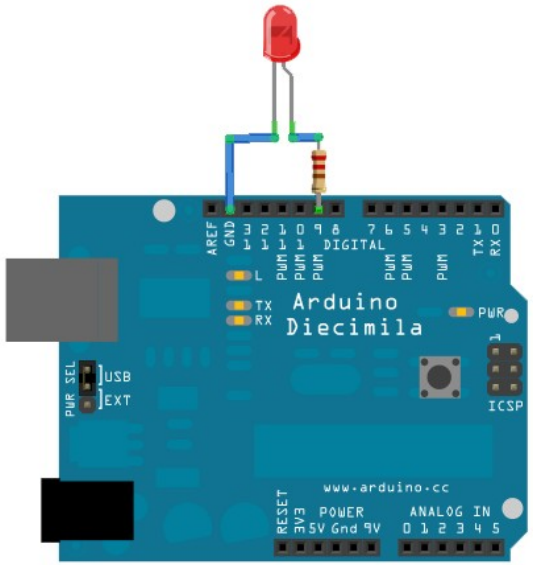
```
const int sensorPin = 0;
void setup () {
    Serial.begin (9600);
}
void loop () {
    int reading = analogRead (sensorPin);
    float voltage = reading * 5.0 / 1024.0;
    float temperature = (voltage - 0.5) * 100;
    Serial.println (temperature);
    delay(1000);
}
```

Analog input peripherals

- The most interesting family of peripherals
  - Dozens of cheap sensors:
    - 3-axis accelerometer (uses 3 analog inputs)
    - Alcohol gas sensor
    - Carbon monoxide sensor
    - Optical dust sensor
    - Flex sensor
    - Force sensitive sensor
    - Vibration sensor
    - Gyro-sensor (2 axis → uses 2 analog inputs)
    - Proximity sensor (both infrared and ultrasonic)
    - Temperature
    - Humidity sensor
    - Etc.
- They convert the physical quantity to analog signals

PWM output peripherals

- **Example**:
  LED fading



- **Code**:

```
const int ledPin = 9;
void setup ()  {
}
void loop ()  {
    for (int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) {
        analogWrite (ledPin, fadeValue);
        delay (30);
    }
    for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) {
        analogWrite (ledPin, fadeValue);
        delay (30);
    }
}
```

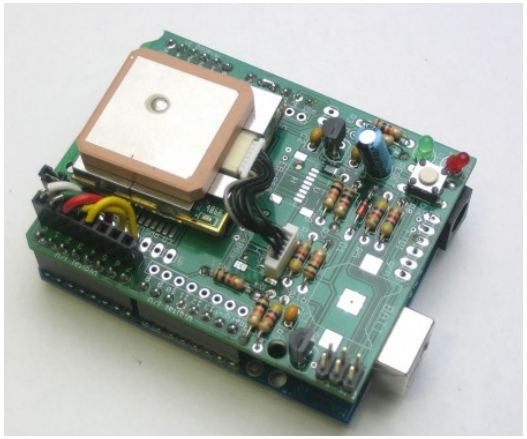
Peripherals connected through a serial port

- Peripherals that can be connected through serial ports:
  - RFID modul: sends the card ID through a serial port
  - GPS modul: sends the geospatial position through a serial port in regular intervals
  - GSM/GPRS modul: can be controlled through serial ports, the received data and the data to transmit is sent on the serial line as well
  - Etc.

Shields

- Complete peripherals stacked over the Arduino
  - There are dozens of shields available:
    - GPS, LCD controller, SD card reader, WIFI, Bluetooth, ZigBee, GSM, ...

GPS Shield
€18



MP3 Shield
€25

2.8” TFT and touch sensor
€40

Implementing the door-lock

Door lock

- What components do we need?
  - A display to interact with the user
  - A numeric keypad
  - A card reader – we use RFID based card reader
  - A switch to open the door
  - An Arduino from the family that is the most appropriate for us

The display

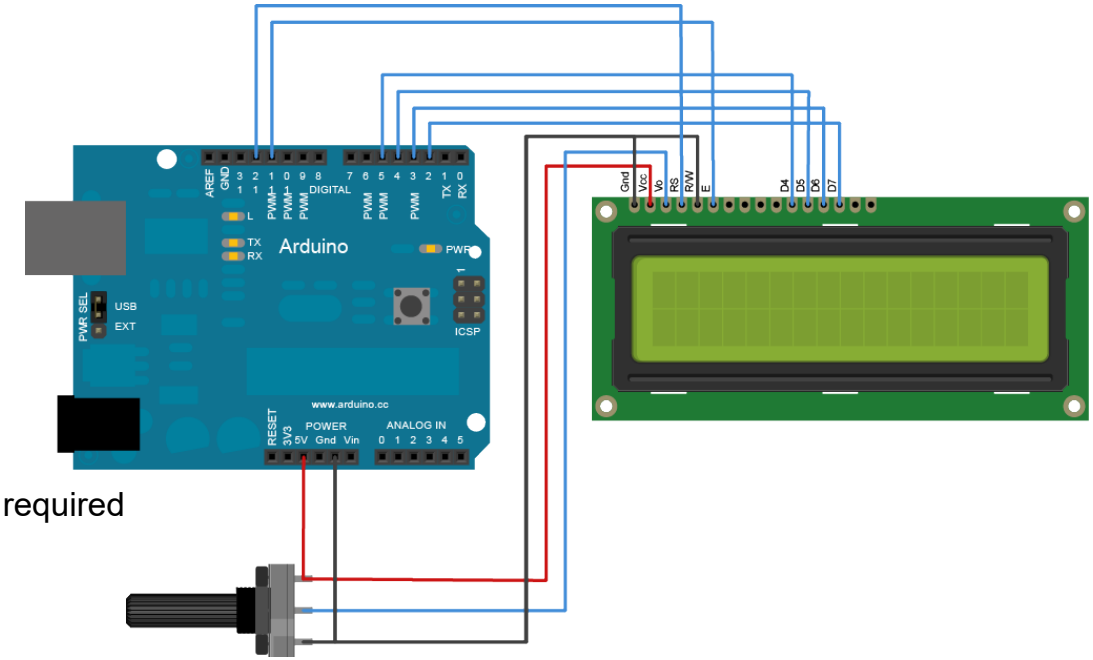
- A cheap display with 2 rows and 16 cols (€8)

- Pins:
  - *Data bus*: D0...D7, but it works in 4 bit mode as well, we use it that way: we connect only D4...D7
  - *RS*: indicates the display if the character sent is a command or a symbol to display
  - *EN*: enable signal, the display samples the data bus when receiving the enable signal
  - *RW*: if we ask something from the display – we dont need it
  - *Power supply* (5V)
  - *Contrast of the display* (potmeter)
  - *Power supply of the backlight*
- We use 6 wires for data transmission: RS, EN, D7, D6, D5, D4

Using the display

- **How to connect it**:
  Google „arduino display”

- **Usage**:
  Instancing the **LiquidCrystal** class
  (write(), setCursor(), blink(), clear(), etc.)



- **Example**:

```
#include <LiquidCrystal.h>
const int numRows = 2;
const int numCols = 16;
// tell it which pins we connected to the 6 wire required
LiquidCrystal lcd (12, 11, 5, 4, 3, 2);
void setup () {
   lcd.begin (numCols,numRows);
}
void loop () {
   for (int thisLetter = 'a'; thisLetter <= 'z'; thisLetter++) {
      for (int thisRow = 0; thisRow < numRows; thisRow++) {
         for (int thisCol = 0; thisCol < numCols; thisCol++) {
            lcd.setCursor (thisCol, thisRow);
            lcd.write (thisLetter);
            delay (200);
         }
      }
   }
}
```

The numeric keypad

- A 4x3 matrix of keys (€3)

- Pins:
  - Has 7 pins, 4 for the rows and 3 for the columns
  - If we push a button, it connects the corresponding column and row lines

Using the numeric keypad

- Google „arduino keypad”
- **Usage**:
  Instancing the **Keypad** class
  (getKey(), waitForKey(), getState(), etc.)

- **Example**:

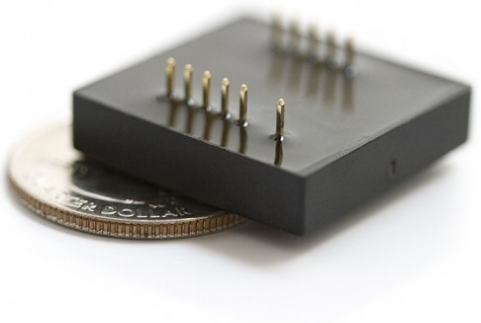
```
#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] = {  {'1','2','3'},   {'4','5','6'},   {'7','8','9'},   {'*','0','#'}};
byte rowPins[ROWS] = {32, 22, 24, 28}; // where did we connect the row pins
byte colPins[COLS] = { 30, 34, 26 };  // where did we connect the column pins

Keypad keyPad = Keypad ( makeKeymap (keys), rowPins, colPins, ROWS, COLS );

#define ledpin 13
void setup () {
   digitalWrite (ledpin, HIGH);
}
void loop () {
   char key = keyPad.getKey();
   if(key)  {
      switch (key)  {
         case '*':
            digitalWrite(ledpin, LOW);
            break;
         case '#':
            digitalWrite(ledpin, HIGH);
            break;
      }
   }
}
```

The RFID card reader

- Radio-frequency identification
- Each card has a unique code consisting 12 hex digits
- Reader: we have an expensive one (€25), because we could find only that one
- Tags: cheap (€1 each, cards, buttons, etc.)



- Communicates through serial port
- Pins:
  - Power supply (5V)
  - External antenna (we use the internal one)
  - Format selection (we use the ASCII format)
  - 2 wires for data transmission (we use only one of them, as a serial line)
  - LED/buzzer at card reading (we dont need it)
  - Reset
- Connects to Arduino with only 2 wires:
  - Reset (to digital pin), D0 (to serial RX pin)

Using the RFID reader

- Google „arduino id-12”
- **Usage**:
  as a serial device

- **Example**: (reset: pin 2, D0: pin RX1)

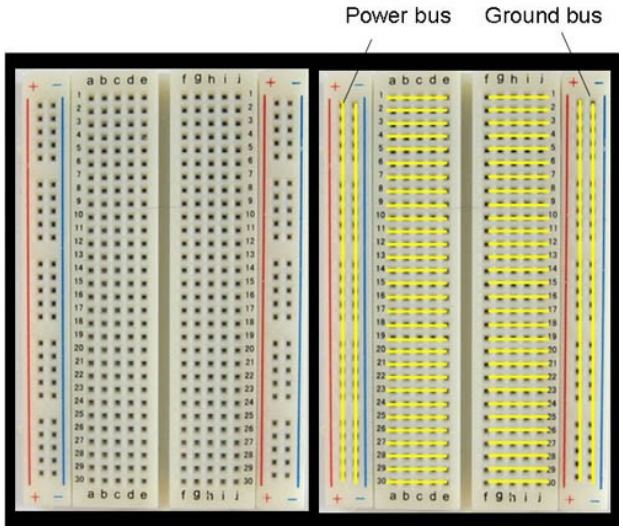
```
const int RFIDResetPin = 2;                 // Reset pin of the reader is connected to digital pin 2
char userID[13];                            // Card ID will be stored in this array
void setup () {
   Serial.begin(9600);                      // set up serial port 0 to 9600 bps (for debug)
   Serial1.begin(9600);                     // set up serial port 1 to 9600 bps (RFID reader connected here)
   pinMode(RFIDResetPin, OUTPUT);           // set RFID Reset pin to output
   digitalWrite(RFIDResetPin, LOW);         // RFID reset, we generate a rising edge. Set it to low...
   delay (100);                             // … wait a bit ...
   digitalWrite(RFIDResetPin, HIGH);        // … raise it to high.
}
void loop () {
   while (Serial1.available()) {            // If bytes arrived from the reader
      int readByte = Serial1.read();        // Read the next byte arrived from the reader
      if (readByte == 2)                    // ASCII 0x2 means „start of message”
         index = 0;                         // Initialize index variable, userID is filled from the beginning.
      else if (readByte == 3) {             // ASCII 0x3 means „end of message”
         userID[index] = 0;                 // Terminate the card ID with a NULL.
         Serial.println(userID);            // Send it to the debug serial port. This appears on the PC connected.
      }
      else
         userID[index++] = readByte;        // We are in the middle of the message, store the character.
   }
}
```

Selecting the proper Arduino

- What to check:
  - How many digital input/output pins we need:
    - For the display: 6
    - For the keypad: 7
    - For the RFID reader: 2
    - Total: 15
    - The Leonardo has 20, but it did not exist when we bought it. Its predecessor had only 14, thus we selected ADK (it has 54)
  - Further input and output pins:
    - We don’t need analog inputs and PWM outputs
    - We need a serial port for the RFID reader
    - If we want to debug, we need another serial port
    - The Leonardo has 2 serial ports (its predecessor had only 1)
- Ideal choice: **Leonardo**

Assembling the hardware

- Using a breadboard, without soldering
  - Breadboard: a set of pre-connected holes:



The software

- We use a state machine with 3 states
  - **start** state: waiting for the card
  - **rfid** state: card is there, the RFID reader is sending the characters of the card ID
  - **code** state: waiting for key press
- The 12 characters of the card ID and the 4 characters of the PIN code are concatenated

  → We obtain a 16 character long string
- If this string equals to one of the pre-stored ones, we open the door

The software - 2

```
#include <LiquidCrystal.h>
#include <Key.h>
#include <Keypad.h>

const int RFIDResetPin = 2;
const int LEDPin = 13;

char userID[17];                                  // the card ID and the PIN code are stored here (concatenated)
int index = 0;
enum estate { start, rfid, code };                // state of the state machine
estate state = start;

LiquidCrystal lcd(49, 47, 48, 46, 44, 42);        // the display has been connected to these pins

const byte ROWS = 4;
const byte COLS = 3;

char keys[ROWS][COLS] = {  {'1','2','3'},   {'4','5','6'},   {'7','8','9'},   {'*','0','#'} };

byte rowPins[ROWS] = {32, 22, 24, 28};            // the row pins of the keypad are connected to these pins
byte colPins[COLS] = { 30, 34, 26 };              // the column pins of the keypad are connected to these pins

Keypad keyPad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

// The accepted card IDs and their PIN codes. First 12 character: card ID, last 4 character: PIN code
char* codes[] = {"010B4CF292261234", "010B4CED58F37899", "010B11C56FB19024", "010B1147E8B41290",
                 "010B11C5F12F7085", "010B112F3F0B0963", "010B11481C4F7412", "010B1148095A3254",
                 "010B1147F3AF6325", "010B114806551589", "010B1147FEA28563", "010B11C56DB33574",
                 "010B4CF0D5637412", "010B4CF26DD96521", "010B4CE9B9164589", NULL};
```

The software - 3

- The mandatory **setup ()** function:

```
void setup() {

   Serial.begin(9600);                  // set up serial port 0 to 9600 bps (for debug)
   Serial1.begin(9600);                 // set up serial port 1 to 9600 bps (RFID reader connected
here)
```

-

```
   pinMode(RFIDResetPin, OUTPUT);       // set RFID Reset pin to output
   digitalWrite(RFIDResetPin, LOW);     // RFID reset, we generate a rising edge. Set it to low…
   delay (100);                         // … wait a bit …
   digitalWrite(RFIDResetPin, HIGH);    // … raise it to high.

   pinMode(LEDPin, OUTPUT);              // set the pin of the built-in LED to output

   lcd.begin(16, 2);                     // or display has 16 columns and 2 rows
   lcd.print("Touch the card!");         // write message to the display
}
```

The software - 4

- The mandatory **loop ()** function:

```
void loop () {
   while (Serial1.available()) {                  // If the RFID reader transmitted some data
      int readByte = Serial1.read();              // Read the next character sent by the RFID reader
      if (state == start && readByte == 2) {      // If we are in the start state and a „start of message” character is received,
         state = rfid;                            // we move to the rfid state
         index = 0;
      }
      else if (state == rfid && readByte == 3) {  // If the „end of message” character is received (ASCII 0x3),
         state = code;                            // we move to the state waiting for the PIN code
         lcd.clear();                             // Clear the display,
         userID[index] = 0;                       // put a terminating 0 to the end of the card ID,
         lcd.print(userID);                       // print it to the display,
         lcd.setCursor (0,1);                     // set cursor to the beginning of the second line
         lcd.print("Enter code");                 // print a message
      }
      if (state==rfid && readByte != 2 && readByte != 10 && readByte != 13 && index<12)     // if card reader keeps sending card ID
         userID[index++] = readByte;              // store it
   }

   char key = keyPad.getKey();                    // check if there was a key press
   if (state == code && index == 16) {            // If we are waiting for a key, and we got the last digit (we have the 12 + 4 characters)
      userID[index] = 0;                          // put a terminating 0 to the end
      Serial.println(userID);                     // print it to the debug serial port
      checkCardAndCode();                         // check if it correct, and open the door if it is correct
      state = start;                              // go back to the initial state
      lcd.clear ();                               // clear display
      lcd.setCursor (0,0);                        // move cursor to the upper left corner
      lcd.print("Touch card!");                   // print message
   }
   else if (state == code && key)                 // if there is a key press, but this is not the last digit
      userID[index++] = key;                      // store it
}
```

The software - 5

- Checking if the card and the PIN code is valid:

```
void checkCardAndCode () {
   lcd.clear ();                          // clear display
   lcd.setCursor (0, 0);                  // move cursor to the upper left corner
   int ix = 0;
   while (codes[ix]) {                    // check all codes stored
      if (!strcmp(userID, codes[ix])) {   // if there is a match
         lcd.print ("OK!");               // print a message
         digitalWrite (LEDPin, 1);        // switch on the LED
         break;                           // we don't have to check the validity any more
      }
      ix++;
   }
   if (!codes[ix])                        // if we reach the end of all stored codes, and still dont find the one given,
      lcd.print ("Denied.");              // print bad news

   delay (1000);                          // wait 1 second

   digitalWrite (LEDPin, 0);              // switch off the LED
   digitalWrite(RFIDResetPin, LOW);       // RFID reset, we generate a rising edge. Set it to low...
   delay (100);                           // … wait a bit …
   digitalWrite(RFIDResetPin, HIGH);      // … raise it to high.
}
```