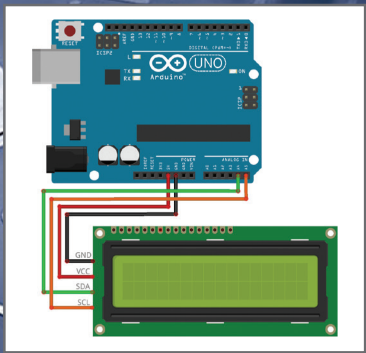
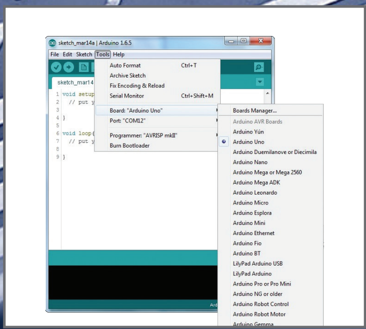
USING THE RFID STARTER KIT

HOME AUTOMATION PROJECTS WITH ARDUINO





Günter Spanner

elektor

LEARN DESIGN SHARE

Introduction 9

Chapter 1 ● A quick introduction to the Hard- and Software 11

1.1 The first Functional Test 14

1.2 Programming the Arduino 16

1.3 Resistors: A Basic Element of Electronics 19

1.4 Are you well connected? Jumper cables 20

1.5 LEDs 21

1.6 Battery power for the Arduino 22

Chapter 2 ● Fire up the Arduino - Projects for beginners 23

2.1 Alarm system simulator 23

2.2 SOS Emergency signal 23

2.3 Mood lighting with a multicolor LED 24

2.4 Switches that Bounce 26

2.5 Data input using a keypad 30

2.6 Warning signals of every pitch 34

2.7 Too few port pins? You need a shift register 37

2.8 Binary counter 38

2.9 Model airfield runway lights 40

2.10 Serial Data Output 40

2.11 Voltage measurement on a row of LEDs 42

Chapter 3 ● Display Technology 46

3.1 An LCD for the Arduino 46

3.2 Attention! Warning Signal Display 49

3.3 Driving Seven-Segment Displays 50

3.4 Pin-saving Shift Register 55

3.5 Universally useful: The 4x7 segment Display Module 57

3.6 A Counter… Not just for beans 59

3.7 The SevenSeg Library 63

3.8 A Digital Clock 64

3.9 For Numbers, Characters and Icons: 8x8 Dot Matrix Display 64

3.10 The Running Light Spot 67

3.11 Beaming Smileys and Glowing Icons 68
3.12 Mini Display using an LED Matrix 71
3.13 A Joystick-controlled light point 74
Chapter 4 • Measuring the Environment 77
4.1 Dry Out the Cellar: Hygrometer Monitors Humidity. 77
4.2 Weather Station with LC Display 80
4.3 Workplace Bright Enough? A Digital Lux Meter. 82
Chapter 5 • Sensor Technology 86
5.1 The Flame Detector 87
5.2 Alarm system with tilt switch 90
5.3 Precise Temperature Measurement with the LM35 Sensor. 92
5.4 Measuring low Temperatures. 94
5.5 Shout or whisper - measure the level with a sound sensor 95
5.6 Remote Control without a Transmitter: The intelligent Clap Switch 98
5.7 Rain or shine? A water level sensor can alert you 100
5.8 Umbrella alert! – A rainfall alarm. 102
Chapter 6 • Motors and Servos control the world 104
6.1 The Stepper Motor and Motor Driver Module. 105
6.2 From rotation to single steps. 107
6.3 Turntable to display Jewelry or scale models. 109
6.4 Use the Joystick for Motor Control 109
6.5 The Servo as a Universal Actuator 110
6.6 Controlled Motor Power: The Servo 110
6.7 The Servo Library 113
6.8 Precise Servo Control 113
Chapter 7 • Banish the Spaghetti: Control Wirelessly 116
7.1 For Convenience: Remote Control with an IR Receiver 116
7.2 A Remotely Contolled LED 119
7.3 Wireless Data Reading: The RFID module. 120
7.4 Contactless door entry control. 124
7.5 RFID Tags Store Data 126

Chapter 8 • Experimental projects for Advanced Users 129

8.1 Always the Right Time: The RTC Module 129

8.2 Digital clocks and timers for precise time measurements 134

8.3 Conway's Game of Life 137

8.4 Hello Matrix! 140

8.5 Live Ticker and Running Text 143

8.6 Switching High Power Loads: The Relay Module 146

8.7 Remote controlled Halogen Lamp 147

8.8 Keypad Door Entry System 148

Chapter 9 • Principles of Arduino Programming 152

Chapter 10 • Use of Libraries 160

Chapter 11 • Fault finding 162

Chapter 12 • Components and Modules 163

Chapter 14 • Literature 184

Chapter 15 • Illustration Listing 185

Index 188

Introduction

Without doubt the Arduino system has become 'the' basic component amongst electronics communities and makers. Earlier it was necessary to begin with a bare microcontroller as the basic building block of any system, now the Arduino is more and more frequently used. Armed with the Arduino system your first steps into the world of microcontroller technology could not be made any simpler. Instead of a complex programming environment, where you would be expected to start coding from scratch you can now work with a highly intuitive development environment combining pre-tested library routines to get you up and running in no time.

When it comes to plugging in additional hardware to the basic Arduino board the developer is still largely on their own. There is a wealth of special expansion boards for the Arduino but these are often only useful for some specific application areas. If you really want to build some innovative projects you often have to get down to component level. This presents many beginners with major problems.

There are some developer's kits which can help in this area. These contain a range of components that you can use with the Arduino to build lots of different projects and help you understand how they work.

That is exactly the purpose of this book. It explains how a wealth of practical projects can be built from a single kit. This kit, called the 'RFID Starter Kit for Arduino UNO' contains more than 30 components, devices and modules from all areas of modern electronics.

In addition to more simple components such as LEDs and resistors there are also complex and sophisticated modules that use the latest technology such as:

- A humidity sensor
- A multicolor LED
- A large LED matrix with 64 points of light
- A 4-character 7-segment LED display
- An infra red remote-controller unit
- A complete LC-display module

Additional special devices include:

- A servo
- A stepper motor and controller module
- A complete RFID reader module and security tag

With these components you can build a whole lot of different projects. For the beginner, we first start off with a few simple introductory experiments. More experienced users can on the other hand go on to build the more complex projects that are described a bit later in the book.

In addition to precise digital thermometers, hygrometers, exposure meters and various alarm systems, there are also practical devices and applications such as

- A fully automatic plant irrigation system
- A sound-controlled remote control system
- A digital clock with versatile display possibilities
- A multifunctional weather station

plus much more.

The projects presented here could not be classed as ‘laboratory prototypes’ but are instead practical and useful designs, fully described with lots of hints and tips on their use in hobby, household and professional applications.

All the projects can be built using the components supplied in the Elektor kit. Alternatively, most of the components can also be purchased from specialist dealers or from online auction sites. At the end of the book a description of these components and a source from where they can be purchased is given. With this information you can replace defective or lost components.

This also gives you the option to purchase components to build individual projects without the the need for the entire kit. The SainSmart Boards have also been offering a cost-effective yet high-quality Arduino clone for some time now. Their boards are fully compatible with the classic Arduinos.

All the sketches given in this book are available to download from:

www.elektor.com

When the sketch is not identical to the listing in this book the downloaded version will be the latest version and is the one that should be used.

Chapter 1 • A quick introduction to the Hard- and Software

In general, 'Arduino' can be used to refer to the development board itself, i.e. the hardware of which there are now many variants of the original Arduino board but we will go into that in a little more detail below. Arduino also refers to the programming environment, together they make up the 'Arduino System'.

The Arduino hardware consists of a microcontroller board. This is a circuit board or PCB on which there are several other electronic components besides the microcontroller itself.

Along the two long edges of the board the Arduino has a row of header sockets commonly referred to in the text as the Arduino's pins. The pins are used to connect the various electronic components to the Arduino. These components, eg pushbuttons, Light Emitting Diodes (LEDs), various sensors, potentiometers, displays, motors, servos etc. will all be used in the following chapters.
There are now lots of different versions of Arduino boards which can be used with the Arduino software. This includes boards of different size all labelled with the official 'Arduino' or 'Genuino' logo.

In addition to the official series of boards there are a number of unofficial third party boards or clones which are substantially cheaper than the originals. They are largely compatible in terms of both hardware and software which means that electrically they produce the same signals and they can be programmed with the Arduino IDE.

The classic Arduino controller boards have names like:

- Arduino UNO,
- Arduino MEGA,
- Arduino Micro etc.

While their corresponding clones have names such as:

- Seeduino UNO,
- Funduino MEGA,
- Freeduino Micro
- SainSmart UNO etc.

The Arduino UNO (see below) is used throughout this book for all of the projects described here. One important compatible board called the SainSmart UNO, can also be substituted for the original Arduino in all the projects.

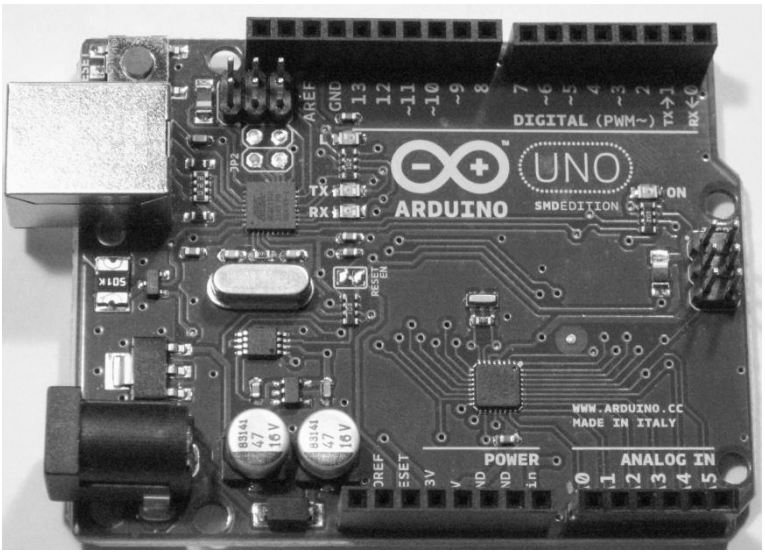


Illustration 1: An original Arduino UNO

In addition to all the electronic components in the kit there are also a bunch of jumper wires that are used for making fast and flexible connections between the Arduino and other components. The flexible cable has a stiff pin at each end for it to plug into the Arduino header strips. Their flexibility makes it easy connect up large structures. Using these wires eliminates the need for time-consuming soldering. They also plug into the protyping plug board or breadboard that we make use of here to build up more complex circuits.

Their flexibility allows you to test projects and devices without having to make and unsolder joints repeateadly. This construction method has gained broad acceptance in the wider Arduino community.

Once the circuit idea has been tested you might wish to make a more permanent version of the circuit using either perf board or a proper custom-made PCB.

The prototyping breadboard or plug board is a really useful base on which to build up circuits. It's particularly good for teaching and training and once the circuit has been tested and understood all the components and wires can just be unplugged and re-used later. With careful handling the plug board and accessories will last a long time.
The following illustration shows the structure of a breadboard. The lines indicate how the contacts are connected to each other. This arrangement makes the electrical connections between components and jumpers leads plugged into the board without the need for screws or solder.

Chapter 2 • Fire up the Arduino - Projects for beginners

Now you have been introduced to all the components and the Arduino board has already been given a functional test the fun can begin. First off we start with some simple projects which you can build without the need for too much additional hardware. The accompanying sketches should also not cause any problems. They may be simple but working through the examples you will get to build many practical projects that you will find are very useful.

2.1 Alarm system simulator

Even this first simple project has a real practical use that could save you a great deal of stress. It is an alarm system simulator, useful as a burglar deterrent.
The most basic test program for the Arduino uses the blink.ino sketch. It makes an LED regularly flash on and off.
In this sketch called flash.ino the LED remains off for three seconds and then flashes briefly for a tenth of a second. This is a typical signal used by car and house alarm systems to indicate that the system is armed.
The flashing LED can be installed somewhere visible and with any luck it would deter an opportunist thief who would not know if this is a genuine signal from a sophisticated alarm system or just a decoy. Best of all, there's no risk of a false alarm waking you up at night!
The sketch looks like this:

```
// Alarm_simulator.ino

void setup()
{ pinMode(13, OUTPUT);
}

void loop()
{ digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(100);              // wait for a tenth of a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(3000);              // wait for three seconds
}
```

2.2 SOS Emergency signal

A single LED is all you need to generate an SOS signal. The three long followed by three short flashes is the internationally recognized signal to indicate distress. Bright LEDs can be seen from a distance and could be a life saver should you get lost on a mountain walk.

The signal is applied to the Arduino internal LED which is connected to digital pin D13 on the Uno. An external LED can be used connected to pin 12 for example, a 1K series resistor

```
int ledPin = 13; // LED in series with 1k resistor on pin 12
```

must be connected in series with pin D12 and the LED.

```
// SOS.ino

int ledPin = 13; // or external LED in series with 1k resistor on pin 12
int i;     // declare variable i

void setup()
{  pinMode(ledPin, OUTPUT);
}
void loop()
{  for (i=1; i <= 3; i++)
    {   digitalWrite(ledPin, HIGH);
         delay(500);
         digitalWrite(ledPin, LOW);
         delay(500);
     }
    delay(1000);
    for (i=1; i <= 3; i++)
    {   digitalWrite(ledPin, HIGH);
         delay(1500);
         digitalWrite(ledPin, LOW);
         delay(1500);
     }
    delay(1000);
    for (i=1; i <= 3; i++)
    {   digitalWrite(ledPin, HIGH);
         delay(500);
         digitalWrite(ledPin, LOW);
         delay(500);
     }
    delay(3000);
}
```

2.3 Mood lighting with a multicolor LED

As well as the standard range of LED indicator lamps which only light up in one color there is also the so-called multicolor LED. These contain three LED chips in one housing.

They normally contain a Red, Green and Blue LED. They also go by the name RGB LEDs. By controlling the on to off ratio of the signal switching each LED it's possible to produce not only the three basic colors but also all the color nuances in between, all the way up to white light.

When the colors change slowly in a predetermined sequence the effect is usually referred to as 'mood lighting'. Here is a sketch that produces this effect:

Chapter 3 • Display Technology

Displays are in the widest sense the interface between man and machine. Together with input devices such as pushbuttons switches and keyboards output devices are also practically indispensible to the operation of electronic systems. At its most basic a simple LED can be used to indicate a condition. It can be used to indicate a particular operating mode of a device or to alert an operator that some limit value has been exceeded, a bright red LED, for example is an effective indicator of an over-temperature condition.

When you need to indicate additional information more complex displays will be necessary. The Liquid Crystal Display (LCD) is a very popular form of display technology. They can be used to show digital values and use characters to convey easily readable messages.

3.1 An LCD for the Arduino

An LCD can be used as a universal output device for the Arduino. The display included in the kit has the following properties:

- It communicates via an I^2C-Bus – Address range: 0x20 to 0x27 (0x20 is the default)
- White characters on a blue background
- Operating voltage: 5V
- Contrast adjustable using a pot
- Display size: 82 mm x 35 mm x 18 mm

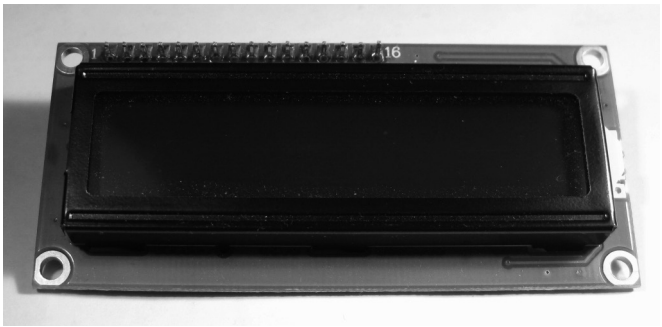


Illustration 29: LCD module

Without doubt LCDs are amongst the most important peripheral components used with microcontroller-based systems. They can display numbers, letters and also various special characters. They will also give your projects a more profession look. The values of temperature, voltage levels and other measurements can be represented accurately on the display as well as more mundane things such as time of day and text messages.

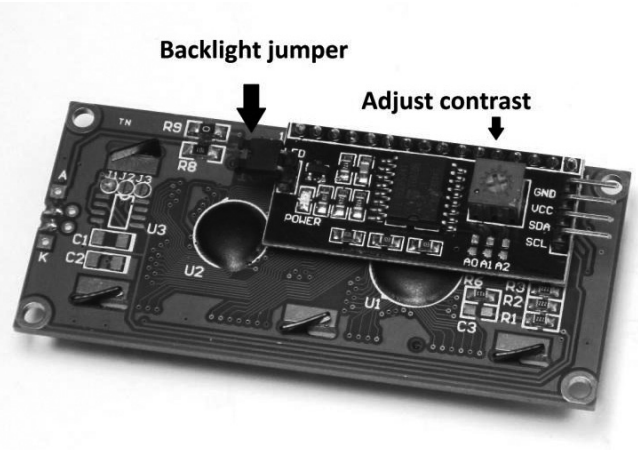


Illustration 30: The LCD module from the back

In contrast to LED dot-matrix displays LCDs are almost exclusively supplied with an integrated controller chip. This is because the dot-matrix LCD has a very large number of pixels needing to be controlled. Even a relatively small 2-line 16 character display is made up of over 1000 pixels. This would far exceed the resources available from an AVR microcontroller. These LCDs therefore form part of a display module with an integrated controller chip. One of the most popular types of controller chips is the HD44780. These can be controlled with just 16 connections. A close look at the display module will reveal the single row of 16 pin header type connections along the top edge of the board.

Connecting all 16 of these pins to the Arduino would use up a lot of the Arduino's I/O capability. There are methods of using the display module which reduce the number of connections required but we would still need at least six I/O pins.

Besides the large display controller chip on the display module there is also a smaller chip which provides the module with an I^2C serial interface so the number of I/O pins required is now reduced to just two for controlling the display. This is an important advantage especially for larger more complex projects which use a large number of Arduino pins. Another difference compared with the standard LCD module is that this one includes a pot to control the display contrast.

If you can't see any characters on the display when you start using it, try twiddling the contrast control pot using a small screwdriver pushed into the central slot in the pot to find the optimal setting.

With the serial interface cabling between the Arduino and display module is now down to just four wires. GND connects to the GND pin on the microcontroller and Vcc with the 5V Pin on the Arduino or Sainsmart Uno board.

The connection labeled SDA on the display module connects to the Uno analog input A4 and SCL to analog input A5. These pins have a double function; they can be used with

analog input signals and also to provide the I²C-Interface signals. The table below shows the four wire hookup between the Arduino and the LCD display module:

LCD module pin	Description	Arduino pin	Arduino function
Vcc	Supply voltage	5 V	
GND	Ground	GND	
SDA	Serial Data	A4	SDA
SCL	Serial Clock	A5	SCL

To use the LCD module with the I²C interface you will need a library which is not pre-installed in the Arduino-IDE. You can download the library from

https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library

and install it in the Arduino IDE.

The following illustration shows how to connect the module to the Arduino:

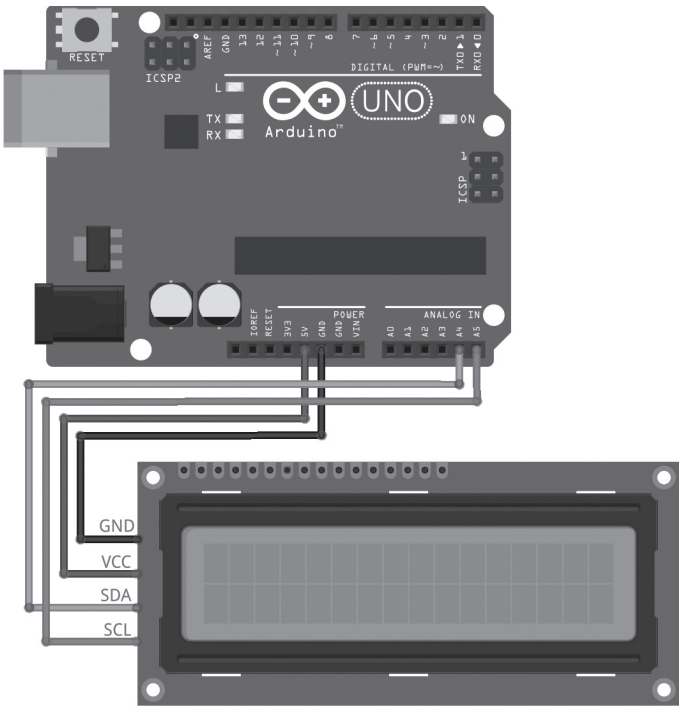


Illustration 31: I2C LCD with the Arduino

Chapter 4 • Measuring the Environment

Monitoring and taking measurements of our environment are some of the most important technological tasks of the modern age. Whether it's tied up with vehicle emissions, protection of the environmental or in the field of robotics. It's becoming increasingly important to have accurate information on the conditions of both our local and remote environments.

4.1 Dry Out the Cellar: Hygrometer Monitors Humidity.

The hardware kit includes a combined temperature and humidity sensor. Along with the two sensors this component also includes the necessary A/D converter to provide both measurements as digital values. This makes it really easy to use. Communication with the sensor including the transfer of all measured values takes place over a single data pin. The sensor unit itself has four pins, one of which is not connected so altogether we use just three connections, two of which are to supply power. The data signal also has a pull up resistor on board.

The two illustrations below show the module and its corresponding pin outs.

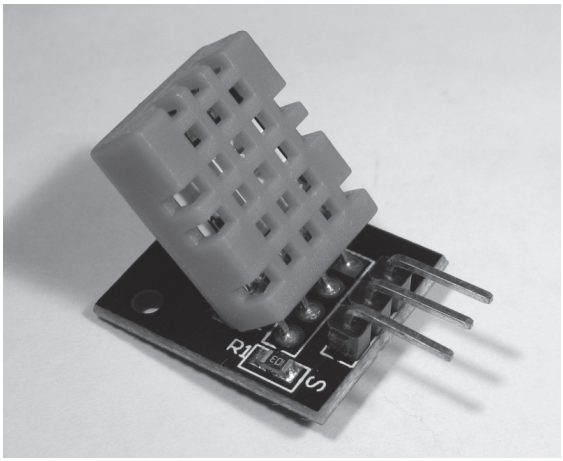


Illustration 49: DHT11Temperature/Humidity sensor

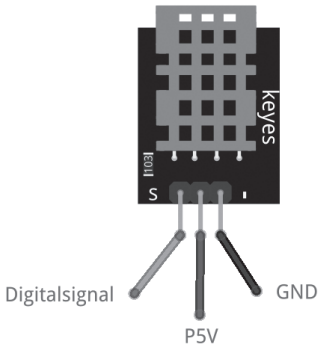


Illustration 50: The DHT11 measures Temperature and Humidity

The following table shows the sensors operating temperature and humidity ranges:

Temperature range	0 - 50 °C
Temperature accuracy	±2 °C
Humidity range	20 - 90 % % Rel. Humidity
Humidity accuracy	±5 % % Rel. Humidity

A suitable library for use with this sensor can be downloaded for free from:

http://arduino.cc/playground/Main/DHTLib

The read() function in this library returns the following values:

- DHTLIB_OK (0): Sensor value and check sum OK.
- DHTLIB_ERROR_CHECKSUM (-1): Check sum error
- DHTLIB_ERROR_TIMEOUT (-2): Time out

Using the library makes it easy to read sensor information:

```
// DHT11_test.ino

#include <dht.h>
dht DHT;
#define DHT11_PIN 4

void setup()
{ Serial.begin(9600);
  // Serial.println(“Type,\tstatus,\tHumidity (%),\tTemperature (C)”);
}

void loop()
{ DHT.read11(DHT11_PIN);
  Serial.print(“Humidity: “);  Serial.print(DHT.humidity,0); Serial.
println(“ %”);
  Serial.print(“Temperature: “); Serial.print(DHT.temperature,0); Serial.
println(“ C”);
  Serial.println();
  delay(1000);
}
```

As you can see the circuit is really simple to hook up. The sensor just connects to the Arduino via three lines. You can choose a different Arduino input pin to transfer the data. To

do this you will also need to change the pin definition in the sketch to the new pin number so that it can communicate successfully with the module:

```
#define DHT11_PIN 4
```

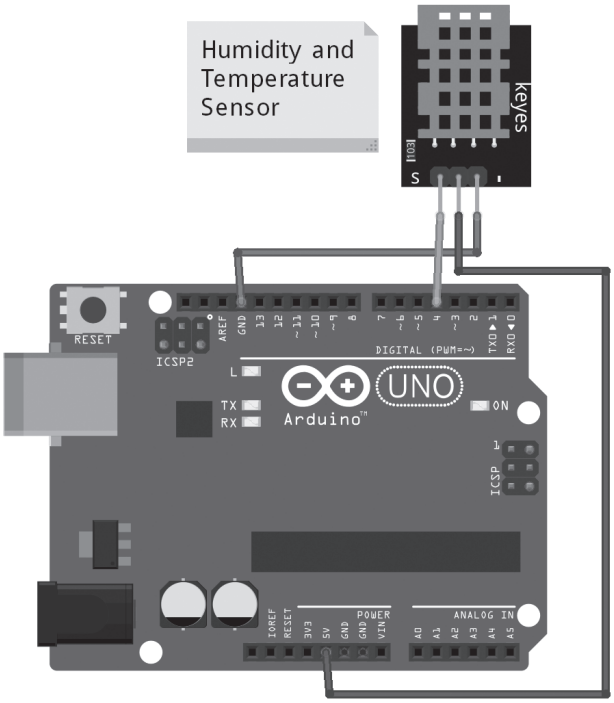


Illustration 51: The Hygrometer provides precise humidity measurements

The measured values can be viewed using the Serial Monitor in the Arduino IDE. To display the values on another PC it will be necessary to run a terminal emulator program like Tera-Term on the PC. A terminal emulator program used to be part of the standard Windows environment but with more recent versions of Windows it needs to be installed. Tera-Term can be downloaded from:

http://ttssh2.osdn.jp/index.html.en

The output data in Tera Term looks like this:

Chapter 5 • Sensor Technology

Sensors or probes are components that are sensitive to certain physical or chemical properties. Important examples of electronically measurable variables are:

- Temperature
- Humidity
- Light intensity
- Pressure
- Sound intensity
- Thermal radiation
- Power or Acceleration
- Magnetism or magnetic field

Sensors are used in every branch of technology. A typical motor car will contain over 100 different sensors. In fact there is hardly any area of high tech that doesn't make use of them. From space travel to medical technology, from smart phones to industrial automation they all rely on information gathered by these sophisticated transducers.
In the previous chapter we've already used sensors to measure light intensity and air temperature. In the following sections we expand the area of application by using some new sensors for

- Flame detection
- Tilt sensing
- Noise level detection

Usually a single sensor is used to make a specific measurement. The value produced must be reproducible and is usually expressed as a voltage level or a change in resistance of the sensor. In general it's preferable for the sensor to produce an output value which has a linear relationship to the measured variable. Some sensors however may have a non-linear characteristic but it's easy to convert the values in the microcontroller software.
Some external activity can introduce inaccuracies in sensor readings. This is an effect known as cross-sensitivity where a sensor's measurements are affected by other unrelated variables. Some examples are:

- Influence of air humidity on electric field strength measurements
- Temperature influences on light sensors
- The influence of mechanical vibrations on sound transducers
- Thermal effects on humidity or pressure sensors

A lot of effort can go into minimizing the influence these unwanted variables can have on the measured values. A common technique is to measure the interfering variable directly and use this value to provide a mathematical correction to the wanted variable. The humidity module we used earlier contains inbuilt temperature compensation so there is no need for the user to make any further correction to its measured values.

It's good practice however to always check to what extent cross-sensitivity can influence the required measurement precision of any sensor you plan to use.

5.1 The Flame Detector

The flame detector is a special type of opto-sensor similar to one we used in a previous project. It is sensitive to sources of infrared light (from around 750 to 1100 nm) and reacts particularly strongly to naked flames. Most of the visible light is filtered by the special dark plastic package surrounding the light-sensing area. This makes the detector look like a black LED and should make it easy to identify. Despite the strong filter the detector can still be influenced by visible light, special measures must therefore be taken in order to ensure it can reliably detect a flame.

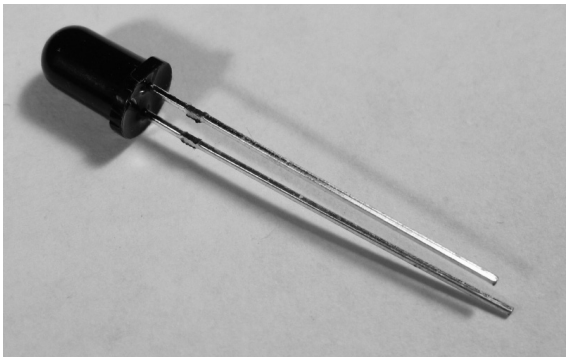


Illustration 57: The flame sensor

Flame sensors are widely used in equipment such as gas boilers where they play an important safety role to quickly shut off gas to the boiler if they detect the flame has gone out. Without this safety mechanism the room would fill with gas and present a serious risk of explosion.

In principle the flame sensor is nothing more than an IR photo diode sensitive to the infra red radiation produced by a naked flame. The sensor has a sensitive acceptance cone of approximately 60° around its central axis.

In the circuit the IR diode is reverse-bias connected with a series 10 Kohm resistor. This configuration makes a voltage divider network producing a voltage level which we read at one of the Arduino's analog inputs. The illustration shows how the flame sensor is connected to the Arduino.

Chapter 6 • Motors and Servos control the world

Stepper motors are much more versatile than standard DC motors. Their stator and with it the axle, can be rotated very precisely which makes them ideal for electromechanical-control applications. They are also widely used in the field of robotics.
The technique required to control a stepper motor (also just known as 'steppers'), however is much more complex than for a simple DC motor. Unlike a DC motor, stepping motors have at least four connections. The number of connections depends on the design. There are basically two main types of motor:

- Bipolar motors
- Unipolar motors

Unipolar motors usually have six connections. Their internal structure is more complex than bipolar motors so for the sake of simplicity we will restrict our study to bipolar motors only. Bipolar stepper motors use two independent coils, each having two connections. This gives us the four connections we associate with bipolar motors. The motor included in the kit has four connections so it must also be a bipolar type stepper motor. The diagram below shows the internal arrangement of the motor.

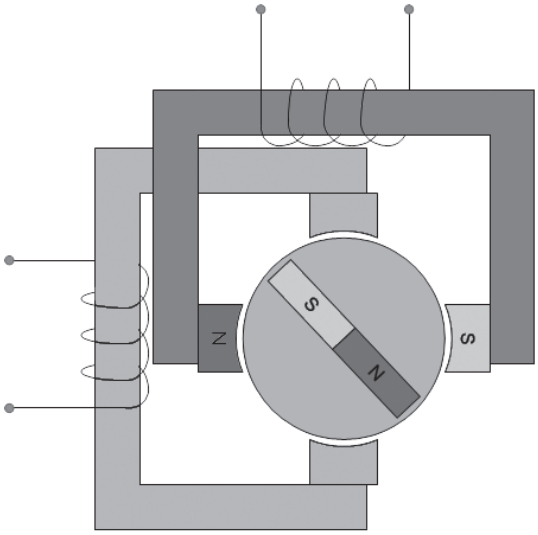


Illustration 72: Bipolar stepper motor basics

To make the motor turn one coil after another is energized. This is made in sequence to produce a rotating field which the magnetic stator in the centre follows. The electrical switching required to produce this rotating field can be easily generated by a microcontroller.

The amount of current that an Arduino port pin can handle is not enough to drive the motor coils directly. The maximum amount of current that each pin can source or sink is 40 mA. The stepper motor coils operate with a current of several hundred milliamps so it's neces-

sary to use a driver chip in between the Arduino and the stepper motor coils to handle the extra current required. A common device used in this type of application is the ULN2003 APG driver chip. This chip can also be found in the motor-driver module in the hardware kit. To make the stepper motor stator turn the microcontroller must output a sequence of 4-bit codes to produce the rotating field. The required signal sequence looks like this where the letters refer to the state of the four outputs connected to the coils.

Step	**A**	**B**	**C**	**D**
1	1	0	1	0
2	0	1	1	0
3	0	1	0	1
4	1	0	0	1

Each time the four outputs change state the stepper motor rotates by one step. When the sequence is included in a continuous software loop the motor spins continuously. The loop index value will then indicate the number of steps. This technique can be used to determine how far a stepping motor rotates with high precision. Since the number of steps for a full revolution is known, it makes it also possible to calculate how many steps a program loop must execute in order to achieve a certain number of motor revolutions.
A stepper motor will only rotate at the point the bit pattern changes, in between changes the stator is held in position by the magnetic forces generated by the coils according to the last bit sequence.

6.1 The Stepper Motor and Motor Driver Module

The stepping motor included in the hardware kit is particularly suited for use with the Arduino system. It can even operate at the same voltage level used by the Arduino so will not require a separate power supply. The stepper motor plugs directly into the motor control board. This supplies the motor with sufficient current. The power for the motor is provided by the module so the digital pins of the Arduino only supply digital switching signals. The plug on the stepper motor just connects to the matching socket on the module. The plug shape ensures that it can't be plugged in incorrectly.
The following table shows the connections between the motor driver module and the Arduino.

Arduino pin	**Module pin**	**Comments**
5V	'+'	External supply can be up to 12 V max.
GND	'-'	External ground
D02	IN4	
D03	IN2	
D04	IN3	
D05	IN1	

NB: Please make sure you use the correct pin assignments otherwise the motor will not operate as intended!

The motor develops high torque even with a 5 V supply. This is because of the built-in gearbox in the stepping motor metal housing. By running it from a higher voltage, the motor power can be increased even further. A maximum of 12 volts may be applied to the driver module's '+' and '-' terminals.
The gearbox also gives the motor very fine motor steps. With this motor one full revolution of the output drive shaft is divided into 2048 individual steps of the motor. This however also has a disadvantage because we can only achieve a relatively low maximum rotational speed.

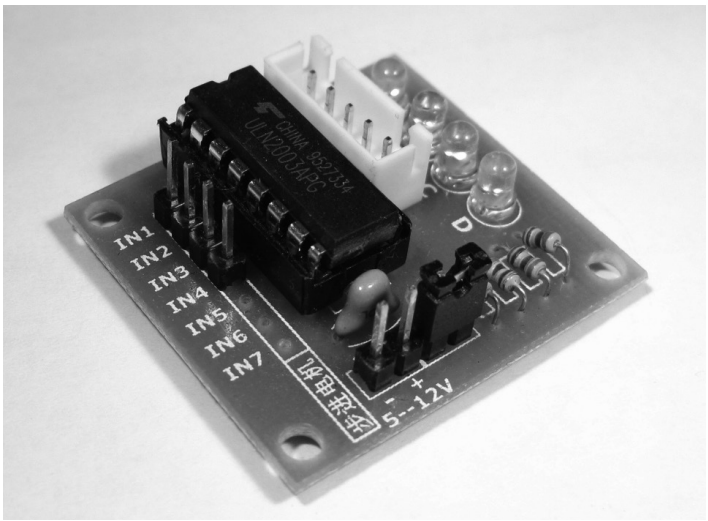


Illustration 73: The stepper motor driver module

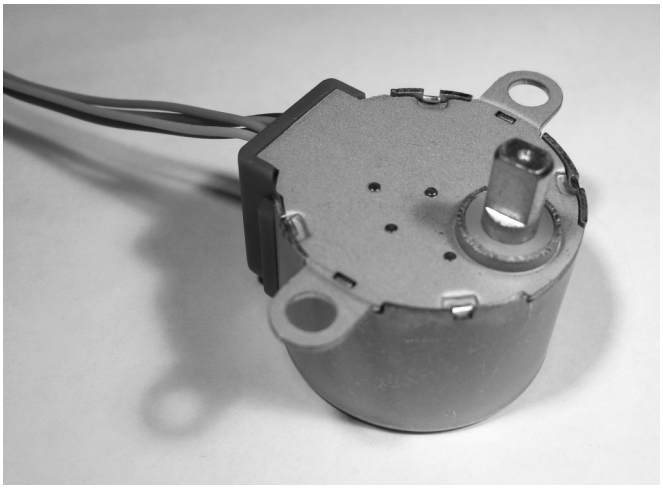


Illustration 74: The stepper motor

Chapter 7 • Banish the Spaghetti: Control Wirelessly

Without doubt one of the most important technologies of today's world is the wireless transmission of signals. Without any visible link, signals can be transmitted over thousands of kilometers. In modern space technology, radio signals even span distances of millions of kilometers. Apart from the use of the radio spectrum, we can also use light signals to transmit information.

In this chapter we go on to investigate both technologies. First signals are sent using invisible infra-red light. This medium is then used here to control ports on the Arduino.

Then we use the RFID module to show how data can be read from a security tag using radio frequency waves. This type of near-field communication is becoming increasingly important in everyday life. From wireless security cards to remotely readable ID tags, we are becoming more reliant on this technology to keep our personal details safe. It is fascinating to learn more about how this technology works.

7.1 For Convenience: Remote Control with an IR Receiver

We have already used a type of IR detector in a flame detector circuit. However, the Arduino can also be remotely controlled via IR commands. In this case, sequences of signals in the form of infrared light are transmitted from a hand-held remote transmitter to a receiver module attached to the Arduino. Here too the signals are not in the visible spectrum.

You may not be able to see them but any digital camera like the one on your smartphone, for instance has an image sensor with a much wider optical bandwidth and is sensitive to IR signals. This is a good way to test your TV remote to see if the batteries are dead. Point it at your camera lens and press a button, you will be able to see on your camera or smartphone screen the LED on the front of the remote controller flashing. The IR LED is flickering because it's sending out a message which contains information on which key you are pressing. The IR receiver (in the TV) converts the flashing sequence into electrical signals which are then decoded. These coded sequences from the hand-held remote controller can also be easily decoded by the Arduino.

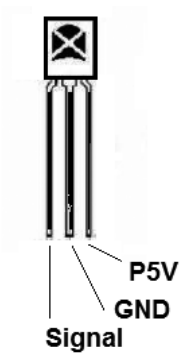


Illustration 81: IR receiver pinouts

Illustration 82: The IR receiver

Connections between the Arduino and IR receiver module are given in the table:

IR receiver pin	**Arduino pin**
P5V	5V
GND	GND
Signal	D2

The couch potato's friend; the IR remote controller transmitter:

Illustration 83: Hand-held IR remote controller

A library of routines which we need to decode the signals is available at

https://github.com/shirriff/Arduino-IRremote

where it can be downloaded and incorporated into the Arduino IDE libraries. Thanks to the library routines, this next sketch is really very short but it's a good start to quickly test out wireless information transmission.

```
// IR_remote_control_test.ino

#include <IRremote.h>

int RECV_PIN = 2; //ir receiver @ pin2

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{ Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
}

void loop()
{ if (irrecv.decode(&results))
  { Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
  delay(100);
}
```

Pressing button '1' in the remote controller shows the number sequence '16724175' in the serial monitor window. This is the number sequence identifying pushbutton 1. All the other keys also have a sequence associated with them which you can discover by pressing the key.

When you press and hold a key the code sequence '4294967295' is eventually received. This code just identifies that a key is held down; any key generates this code when it's pressed continuously.

This table gives the main characteristics of the IR receiver chip:

Supply voltage	**2.7 V to 5.5 V**
Carrier frequency	37.9 kHz
Acceptance angle	90 ° approx.

Chapter 8 • Experimental projects for Advanced Users

Following on from the previous projects which were simpler applications using individual modules and components we now present in these last sections, more complex devices and projects.
These projects typically use several modules combined together so that to start off a real-time module with an LCD builds a really practical digital clock.

We have used some of the components before but the applications here will be more complex such as a running text display or implementing a familiar computer game.
Finally we get to build a controller for halogen lights using a relay module and then a complete door-entry system for a room or cupboard with a keypad to enter the access code.

The projects worked through here are of course just the tip of the iceberg. Using the components supplied with the kit you can build a whole host of useful devices. After working through the projects in this book you will be familiar with all the individual modules and components and well equipped to start using them for your own applications. Here are just a few suggestions:

- Noise-triggered halogen lamp
- A cat flap with RFID recognition
- Mood lighting controlled by a joy stick
- Automatic room lighting brightness control
- Etc, etc…

8.1 Always the Right Time: The RTC Module
The Arduino has its own 16 MHz crystal on board to provide the processor clock signal, this is accurate enough to use also as a time base for a digital clock without any additional hardware. Whenever power to the system is turned off the clock would however lose the time setting and would need to be reset to the current time on power up. One way round this problem would be to continually power the Arduino using a backup battery but an even simpler solution is to use an RTC (Real Time Clock) module. These modules keep track of the time and date powered by their onboard 3 V coin cell.
The RTC module only needs to be connected to the Arduino to provide a high-quality digital crystal timebase. The accuracy of this timebase is very high; the RTC module uses a special watch crystal, the precision of which is slightly better than a standard 16 Mhz crystal.
The RTC module included in the kit uses the popular DS1302 timer chip. A library of functions for this device can be found at:

https://github.com/msparks/arduino-ds1302

The following table shows the hookup of the five connections between the RTC module and Arduino:

RTC pin	Arduino pin
Vcc	5V or D03
GND	GND or D04
CLK	D05
DAT	D06
RST	D07

Here is then RTC module showing the coin cell:

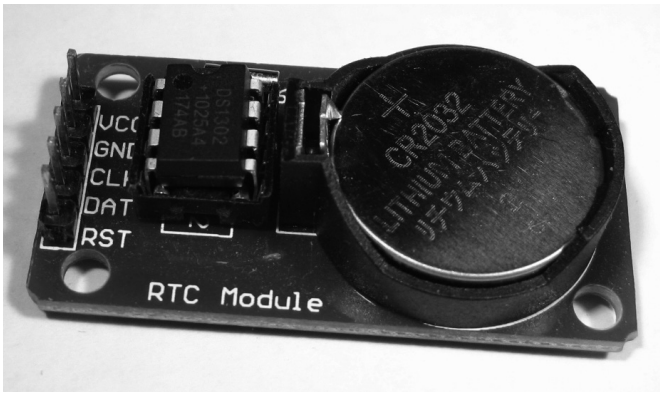


Illustration 90: The RTC module with 3 V coin cell

In theory the module can be wired to Arduino using male/female jumper leads but in practice the wires can be a source of trouble for data transfers. It's better to keep the connections as short as possible or better still plug the module directly into the header sockets on the Arduino. There are two possible ways this can be achieved as shown in the illustrations below.

You can choose whichever configuration is more suitable for your application. If you need to use the space above the Arduino to make other connections then use the first version. If you want the smallest physical assembly then choose the second version.

Chapter 9 • Principles of Arduino Programming

One of the main reasons for the success of the Arduino platform is its super-simple programming environment. The Arduino IDE (**I**ntegrated **D**eveloping **E**nvironment) is very easy to use, even youngsters with no previous experience can have their own programs up and running in no time.
In addition the IDE supports libraries of routines for almost every application. This facility makes it much easier to get your own project working quickly. It's not necessary to start from scratch and reinvent the wheel every time, you can select from a wide range of tried and tested library routines written by other programmers.

For building your own projects it will undoubtedly be helpful to have some background understanding of the Arduino programming language. A brief introduction covering the most important commands is therefore included in the following chapters.

This introduction is not intended as a basic course but as a reference which should help to explain any parts of a sketch which are not immediately clear.

For a more comprehensive introduction check out the bibliography (chapter 14).

The basic program structure
Any Arduino program or 'sketch' consists of two sections:

1. void setup() { instructions; }

2. void loop() { instructions; }

The **setup** is executed first when a program starts running; this section is only executed once and serves to initialize things like the GPIO pin modes and to set up a serial communication channel for example.

After the setup() function comes the loop() function. This section contains the main part of the program code. This is executed repeatedly in an endless loop.

Functions
A function is a block of code which has its own name. When the function is called by its name the instructions contained within it will be executed. Also void setup() and void loop() are nothing more than special functions provided by Arduino's programming system.

Creating your own functions is useful to simplify repetitive tasks and to improve the readability of the program structure. First off you will need to define the type of function. This is determined by the data type of the value returned by the function. For example if the function returns a value which will be an integer, the function type is 'int'. When the function does not return any value the function type is 'void'. After the function type is defined the function name comes next then curly braces or brackets:

```
Typ FunctionName(parameter)  {    instructions;  }
```

Syntax elements

{} curly braces

Curly braces are used to denote the beginning and end of functions, loops and conditional statements.

; Semicolon

A semicolon marks the end of a statement.

/*... */ A comment block

Multi-line comments are text areas that contain no program code but only descriptions of a program section. They begin with /* and end with */. They can extend over as many lines as necessary.

// one-line comment

Comments that just take up just one line begin with a double forward slash //.

Variables and Variable declaration
Variables can be assigned different values during program execution, unlike constants which keep the same value during program execution.

A variable must be declared before it is used. A initial value can also be assigned to it.

```
int value1 = 0;
```

The Arduino system includes some reserved constant names:

HIGH/LOW	Defines pin output state
INPUT/OUTPUT	Defines pin as input or output function
true/false	false = 0, true = not false

Data types
The following data types are available in the Arduino-IDE:

boolean	*TRUE or FALSE*
char	-127 to 127
unsigned char	0 to 255
byte	*0 to 255*
int	-32,7678 to 32,767
unsigned int	0 to 65,555

Chapter 10 • Use of Libraries

The availability of libraries or LIBs and their ease of use are an essential reason for the success of the Arduino system. LIBs are not a new idea; they have been around a long time in other programming environments also. The earlier versions usually incurred a license fee and were not generally known about, whereas the Arduino libraries are free.
It's fair to say that with the advent of the Arduino IDE, this situation changed fundamentally. A variety of standard libraries are already provided free of charge when you install the IDE. More importantly, however, a whole host of various LIBs have been developed by the world-wide community of Arduino users. These libraries are always available free of charge on the Internet.
Thus, a comprehensive set of applications is now available to the general public. In most cases there is a suitable LIB for every planned project or program application.

In most cases, the library can be loaded as a ZIP file from the Internet.
To do this, the ZIP file is first unzipped then the new file directories are copied into the directory using this path:

arduino-x.y.z\libraries

After a restart the library will be installed and ready to use.

Alternatively you can just go to the tab Sketch -> Include Library and Add .ZIP library… then find the file location and double click on the ZIP file to include it. The library manager is also available to check which libraries are installed and also to install additional libraries. The following is a list of all the libraries used in this book. Sometimes it's possible that a link is no longer active. In this case it is usually easy to find an alternative by using a search engine.

Keypad library:

http://playground.arduino.cc/Code/Keypad#Download

LiquidCrystal-I2C-library:

https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library

Seven-segment display:

http://sim.marebakken.com/SevenSeg.zip

Time library:

https://github.com/PaulStoffregen/Time

Chapter 11 • Fault finding

I guess you're here because a project does not work as expected; this section may help. The following points summarize the most common causes of error. Take a methodical approach and go through, ticking them off one after the other. By the time you have finished there a good chance you will have tracked the problem down.

- Some components will only work properly if they are connected the right way round. Check all these 'polarized' components especially the electrolytic capacitors, IC's and LEDs.
- Ensure there are no accidental shorts between connection wires or component leads.
- Double check that all the resistors are the correct value. The color code bands are sometimes difficult to read under poor or artificial lighting. 1 Kohm and 10 Kohm resistors are often incorrectly identified. A digital voltmeter with a resistance range is useful here to check a resistor value.
- When power is supplied from batteries install a new set to make sure they are up to the job. As the supply voltage drops too low it can produce unusual and unreliable operation.

For building Arduino projects the simplified overview diagrams shows what plugs in where and how everthing connects together. This works well when you are just getting started but for larger layouts with several modules it can be a bit confusing. It sometimes helps if you revert to standard circuit diagram symbols and then draw in the connections. This is a kind of reverse-engineering approach which will give you a better understanding of the circuit and help spot a wrong connection.

Chapter 12 • Components and Modules

This book is principally designed to be used together with the RF-ID kit. All of the projects and experiments can be carried out using components supplied in the kit. There may be some people particularly interested in building just a few of the projects described here.

In this case it's possible to order the components needed for the projects of interest. In this next chapter we identify the components and suggest where they could be obtained.

The descriptions also help you track down suitable replacements if any item gets lost or becomes defective.

Breadboard
Standard prototyping breadboards are available in several different sizes. The one supplied with the kit is relatively large with lots of room to build circuits.
The smaller boards are also useful for less complicated projects. The boards are available from online stores and also high street component stockists.

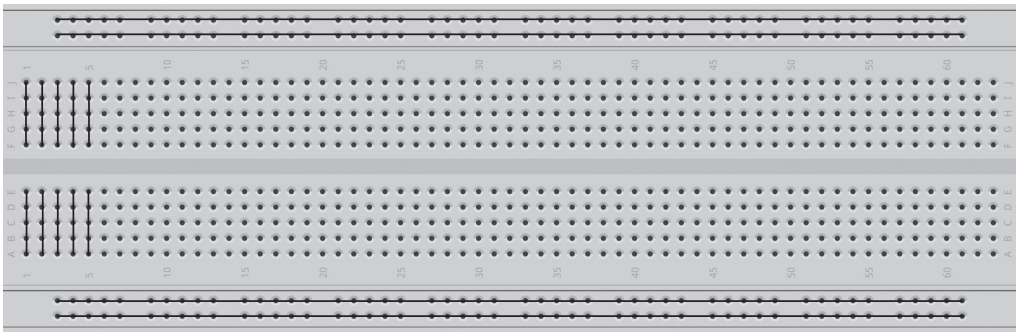


Illustration 98: A protyping breadboard

Resistors
Resistors are available form all stockists of electronics parts. The kit is supplied with:

- 10 x 220 Ohm resistors,
- 10 x 1 K-ohm resistors,
- 10 x 10 K-ohm resistors

It often works out cheaper to buy a bulk pack containing all the standard values rather than buying individual values.

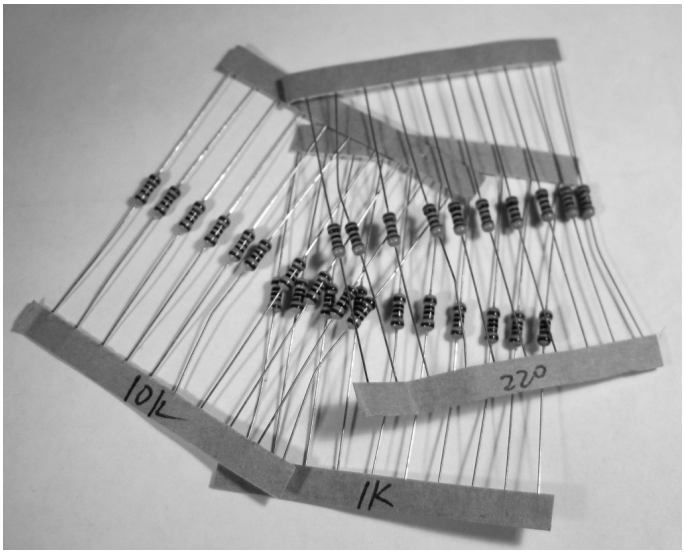


Illustration 99: An assortment of resistors

Arduino

The Arduino board is the main component used in all the projects described in this book. A genuine original board is high quality and therefore relatively expensive, there is also a wide choice of clone-boards available online.

The range of Arduino-clone boards available form SainSmart are low-priced, good quality and have proved to be reliable.

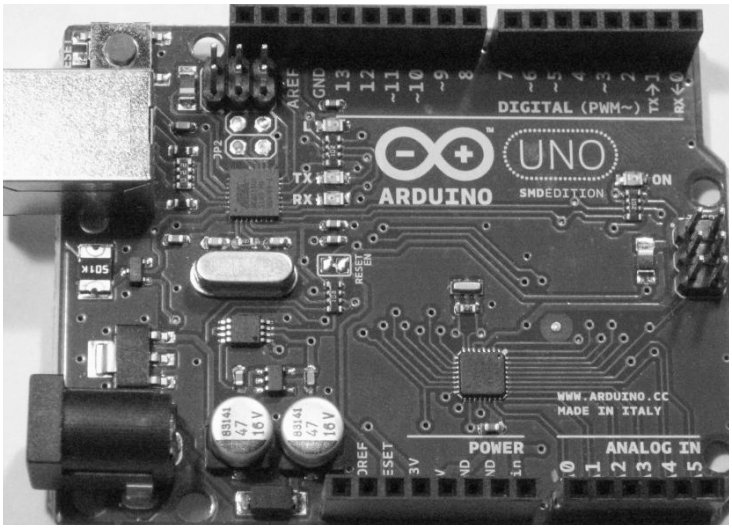


Illustration 100: The Arduino Uno

Index

Symbolen
4x7-segment display 176
74HC595 182
74HC595 IC 37
5161AS 175
{} curly braces 153
// one-line comment 153
; Semicolon 153

A
Alarm system 23
Arduino 11, 164
Arduino IDE 152
Arduino MEGA 11
Arduino Micro 11
Arduino UNO 11
arithmetic operators 154
Array 154

B
Battery 22
Battery clip 180
Bibliotheken 160
Binary counter 38
Bipolar motor 104
BPW40 178
Breadboard 13, 163

C
candle flame 89
Clap Switch 98
Component supplier 182
compound operator 154
Conrad Electronics 183

D
Data types 153
debouncing 29
DHT11 77
DHT11Temperature/Humidity sensor 77
Digital Clock 64
door entry control 124

F
Fault finding 162
Flame Detector 87
Freeduino Micro 11
function 152
Functional Test 14
Funduino MEGA 11
Funktionen 152

G
Game of Life 137

H
Halogen Lamp 147
HD44780 47
Humidity/temperature sensor 171
Hygrometer 79

I
IDE 152
Installer 17
Integervariablen 154
Integrated Developing Environment 152
IR receiver 181
IR Receiver 116
IR remote controller 181

J
joystick 74
Joystick 169
Jumper 20
Jumper cable 166

K
Key fob tag 168
keypad 30
Keypad 169
Keypad Door Entry System 148

L
LCD 46
LC Display 167
LED 21
LED level indicator 44
LED matrix 176
Libraries 160

Light Emitting Diode 21, 165
Live Ticker 143
LM35 92
LM35DZ 182
Lux Meter 82

M
Matrix module 65
microphone 96
Microphone 174
Motor Driver 105
Mouser Electronics 183

P
photoresistor 82
Photo resistor 178
Piezo mini loudspeaker 35
Piezo speaker 177
Potentiometer 174
Program flow control 155
Programming 16
Pulse Width Modulation (PWM) 110
pushbutton 26
Pushbutton 179

R
rainfall alarm 102
rain sensor 103
Real Time Clock 129
Relay module 173
Relay Module 146
Remote Control 98, 116
remote controller 117
Resistors 19, 163
RFID module 167
RFID (Radio-Frequency IDentification) 120
RGB LED module 172
RTC module 170
RTC Module 129
Running Light 67
Running Text 143

S
SainSmart UNO 11
Seeduino UNO 11
Sensors 86
Serial Communication 158
Serial Data Output 40
Serial Monitor 33
Servo 110, 177
Servo Control 113
Servo Library 113
setup 152
SevenSeg 63
Seven-segment display 175
Seven-Segment Display 50
SH1388ASR 176
SH5461AS 176
Shift Register 55
Sketch 18
SN74HC595N 37
sound sensor 96
stepper motor 104
Stepper motor 172

T
Temperature Measurement 92
tilt switch 90
Tilt switch 178
Timing control 157
Transmitter 98

U
UID 126
Unipolar motor 104
Universal Actuator 110
USB cable 13, 165

V
Variable 153
Variablen 153
void loop 152
void setup 152
Voltage measurement 42
VS1838B 181

W
water level sensor 100
Water level sensor 171
Weather Station 80
White card 168