

RFID starter kit for the Arduino Uno

Temperature sensing and door entry system



Using the RFID starter kit from Elektor provides a good basis to carry out numerous electronic experiments with the Arduino Uno. We show just how simple it can be to realize an application.

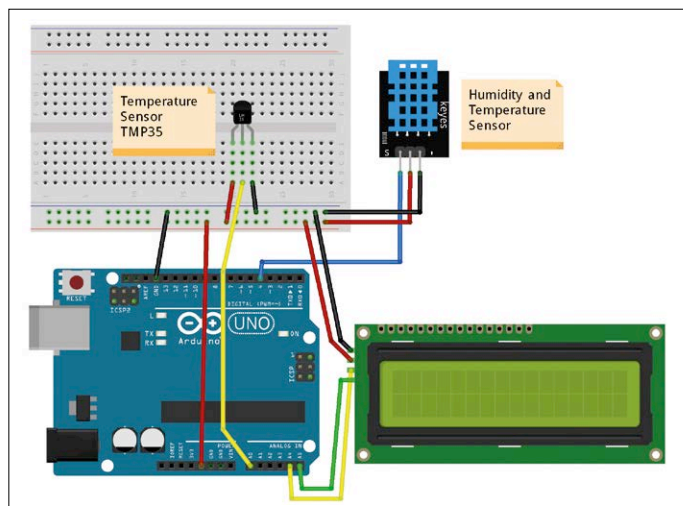


Figure 1. The weather station with two sensors and a display.

The Arduino system or one of its many clones is now often the keystone in many projects built by engineers and makers. Not so long ago a lone microcontroller was the basic building block for any controller-based design, nowadays it's an Arduino board. It has been responsible for introducing computer technology to a whole new generation of users. Instead of needing to get to grips with a relatively complex programming environment, today's makers and electronics tinkerers have the luxury of intuitive development tools in an integrated development environment that is relatively easy to dive into even for a complete beginner.

The situation is a little different from the hardware standpoint. Here developers find themselves largely out on their own. There is of course a wide choice of different shields that plug onto the Arduino board to expand its capabilities but they are often designed for one specific application. To develop new innovative projects it's often necessary to resort to interfacing discrete electronic chips and devices to the controller. This can be a problem for newbies.

For help here we can find some kits containing all the components required to build some projects. Kits like this were popular in the 70s and 80s, at that time you could build things like a siren or a basic medium wave receiver; nowadays we have the added versatility of microcontroller-based projects. One interesting example of the new builder's kits is the 'RFID Starter Kit for Arduino Uno'. This comes in a handy case and contains over 30 state-of-the-art components, devices and modules. The name of the kit is a bit misleading because although it contains an RFID receiver module along with two RFID tags in the form of a credit card and key fob the case is an Aladdin's cave with loads of other useful components.

To begin you will need an Arduino Uno along with the starter kit which amongst other things contains:

- A humidity sensor;
- A multicolor LED;
- A large LED-Matrix with 64 LEDs;
- 4 x 7-segment LED displays;
- A handheld IR remote controller plus IR receiver chip;
- A complete LC-Display module with I2C bus interface.

A more comprehensive list of all the kit contents can be found at [1]. What we go on to describe here is just two example

applications that can easily be built using this box of goodies. The wide range of peripherals included in the kit ensures the number of different experiments and applications you can build Universal weather station with LC-Display.

In the first example application we interface a number of different sensors to the Arduino board to measure environmental data. The DHT11 temperature/humidity sensor is used as well as the LM35DZ temperature sensor. This allows us to build a system to measure and display the values of indoor temperature and humidity as well as the outdoor air temperature. An LC display is used to display the values. The wiring of all the hardware is shown in **Figure 1** and the sketch is given in **Listing 1**.

The **LM35 Sensor** is calibrated during manufacture to supply an output voltage characteristic of 10 mV/°C. In addition it outputs 0 V at a measured temperature of exactly 0.0 °C. The output voltage level will be measured by an ADC with 10-bit resolution (is outputs a value in the range of 0 to 1023). The LM35 has a 5 V supply so the measured temperature is given by the formula:

```
temp = (5.0 * analogRead
(tempPin) * 100.0) / 1023;
```

This however will give quite a poor resolution for the measured temperature range. The LM35 generates a full range output voltage from 0 V to just over 1 V. If we use 5 V as the reference voltage for the ADC it means that 80 % of the available input range will never be used and the measurement resolution of the limited range we are using will be quite poor. To get maximum resolution we can set the internal reference voltage to 1.1 V. With a reference of 1.1 V the formula for temperature needs to be changed. Now the measurement resolution is equal to 1.1 V divided by 1023 which comes out at 0.001075 V or 1.0752 mV. With the output characteristic equal to 10 mV/°C the resulting conversion factor equals:

```
float TempCal = 0.1075; //1.075
/ 10
```

The instruction in the code to give the temperature is:

```
tempLM35 = (LM35val * TempCal);
```

To enable the internal 1.1 V reference voltage source we use the Arduino instruction:

```
analogReference (INTERNAL);
```

Now using this reference voltage the system achieves a resolution of one tenth of a degree Celsius. The reference voltage will of course be subject to tolerances which can be compensated for by adjusting the multiplication factor 0.1075. Once an exact value is determined you can expect the displayed measurements to be accurate over a long period of time because the reference voltage will only drift by a small amount.

With this setup we can measure temperatures from 0 to 110 °C. One tenth of a degree is represented by 1 mV output change and it should also be noted that heat produced by the controller and associated circuitry can be transferred via the sensor leads and will influence the measured air temperature. For use as a room thermometer it's sufficient to display the value to the nearest whole degree but for applications requiring more accuracy don't forget to consider these effects.

The combined **Temperature/Humidity sensor type**

Listing 1. Measure environmental conditions with the DHT11 and LM35.

```
// DHT11_LM35_w_LCD_display.ino

#include <dht.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd(0x27,16,2); // LCD address 0x27
dht DHT;
const int DHT11_PIN= 4;
const int A0 = 0;
float tempLM35 = 0;
long LM35val = 0;
float TempCal = 0.1075;

void setup()
{ lcd.begin();
  lcd.backlight();
  analogReference(INTERNAL);
}

void loop()
{ DHT.read11(DHT11_PIN);

  LM35val = analogRead(A0);
  tempLM35 = (LM35val * TempCal);

  lcd.setCursor(0, 0); lcd.print("Ti="); lcd.print(DHT.temperature,0);
  lcd.print(char(223)); lcd.print("C"); // print unit " °C "

  lcd.setCursor(8, 0); lcd.print("Ta="); lcd.print(tempLM35,0);
  lcd.print(char(223)); lcd.print("C"); // print unit " °C "

  lcd.setCursor(0, 1);
  lcd.print("Humidity: ");
  lcd.print(DHT.humidity,0); lcd.print(" %");
  delay(200);
}
```

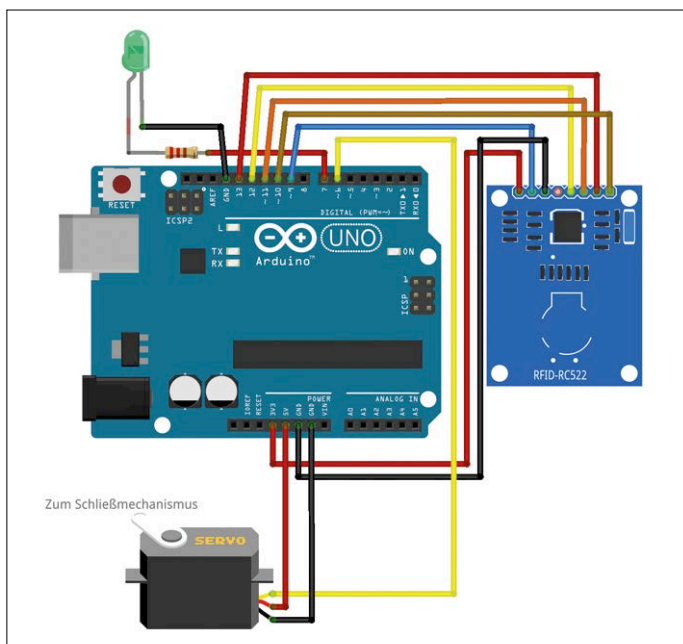


Figure 2. The RFID module plus servo control.

DHT11 communicates using an I2C bus. A complete set of software library functions ensures that reading the measured values is really simple. The **Display** uses the I2C bus and there is a comprehensive set of library functions available also. The function:

```
lcd.print()
```

can be used to display text or measured values. Integrated cursor control ensures problem-free formatting of the values. In this surprisingly compact sketch we use the LiquidCrystal_I2C.h [2], DHTLib [3] and Wire.h libraries. The Wire library is needed for I2C bus communications and like the SPI and Servo libraries (used in the following sketch) are some of the standard libraries included in the Arduino IDE. Both of the other libraries mentioned can be downloaded from the internet free of charge. Should any of the links be dead just use a search engine to find an alternative source.

The weather station can of course be expanded if required by adding more sensors from the RFID kit. The water-level sensor or photoresistor (LDR) can also be used to provide information

to the Arduino. The water-level indicator could tell you if it is raining and the LDR could provide information on light levels which you could then use to record information on sunlight intensity and duration.

Door entry system using RFID security

Another interesting application that can be built with the kit is an electronic door entry system using an RFID tag as a key to open the door. The RFID module included with the kit actually gave the kit its name although it's just one of a bunch of useful items included in the 'RFID Starter Kit'.

Communications to the RFID module take place with the Arduino using the 'Serial Peripheral Interface' (SPI). This type of interface can support communication with many modules all of which use a common Serial Clock (SCK) signal generated by the master controller on the bus to synchronize data exchange. Data is passed using the common Master Output, Slave Input (MOSI) and Master Input, Slave Output (MISO) signals.

In addition to these common signals, each module has its own Slave Select (SS) or chip select signal which is also generated by the master controller when it wants to communicate with the slave module. Each module also has its own reset (RST) input. An Interrupt Request signal (IRQ) can also be generated by a module but we don't use this feature in this application. The RFID module is hooked up to the Arduino as shown in **Table 1**.

Looking at the software we will be using the Arduino SPI-Lib from the Arduino-IDE and the MFRC522-Lib for the RFID module von [4].

With the hardware hooked up according to **Figure 2**, the 'RFID-RC522_data' sketch (**Listing 2**) in the download packet [5] can be flashed to the Arduino. Now when you click on the 'serial monitor' button in the Arduino IDE you will be able to read data sent by the RFID transponder module. When an RFID fob or card is close enough to the coil in the module the data stored in the tag will be displayed (**Figure 3**).

One important parameter which will be used in the sketch is the UID (**Unique ID**), a number unique to the tag. The other information stored in the tag is not used here.

An important application of RFID tags is as a key for door entry control. Authorized personnel will carry a RFID card or key fob which allows them to enter a controlled room. At the entrance will be a RFID reader which only releases the door lock when a tag with a valid UID comes within range. An RFID entry system has a number of advantages compared to systems that use traditional keys. Should a tag get lost, it's easy to delete its UID from the list of authorized personnel and issue a new tag, security will not be compromised and there is no need to replace the lock barrel and issue new keys. The price of a tag is also much less than a metal security key blank.

To recognize a particular UID it's simpler if the hex values stored in the tag are first converted to decimal. This makes it easier to check if the code is valid. The UID is made up of four blocks of hex values and here for simplicity we don't check all

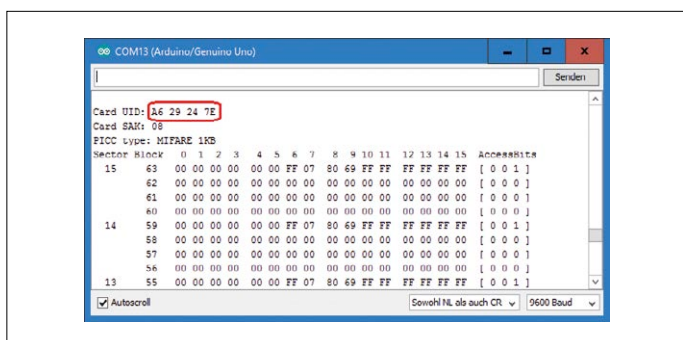



Figure 3. RFID transponder data shown on the Serial Monitor.

of the blocks so the system is less secure than it could be. For maximum security it would be better to evaluate every block but this increases the work involved considerably.

way to build on your existing knowledge and you'll have no excuse not to go on to design and build your own system! 

(160322)

The 'RFID-RC522_servo_lock.ino' sketch for this project is in the download packet for this article. The sketch outputs the UID and its decimal equivalent to the serial monitor. The decimal value can now be used in the declaration:

```
long validCode = 938350; // enter valid code
```

...to assign the card ID to the 'validCode' variable. The system can be used as a door-entry controller; the door will only be unlocked when the system reads a tag containing the correct UID.

A servo motor is included with the RFID kit and this can be used to operate a sliding door bolt. **Figure 2** shows how the servo is hooked up to the Arduino. A green LED is also connected to the board and indicates that the door (or locking mechanism for a locker) is unlocked. The sketch has been expanded at the end to include some basic servo control commands from the servo-library. Make sure that any electronic door entry system you install cannot prevent you from quickly exiting a building in the event of an emergency or power failure.

Summary and outlook

Most modern electronic experimenter kits contain modules that just plug together rather than individual components. This means that we don't get so close to the hardware but reflects the trend in electronics system design so that now we can quickly get modules talking and then decide in software how the system behaves, that can also be quite challenging. Thanks to the wide selection of software library functions we can have a sketch up and running in no time.

These kits are worthy successors to the bags of loose components that were a feature of earlier experimenter's kits. Electronic newbies, practicing engineers and old hands alike are sure to find that these kits interesting. The range of components is sure to get you thinking what you could use them for and they are a good

Table 1. Hook up of the RFID module and Arduino.

RC522-Pin	Pin Arduino Uno
VCC (3V3)	3.3 V
GND	GND
RST	9
SDA (SS)	10
MOSI	11
MISO	12
SCK	13

Web links

- [1] RFID starter kit: www.elektor.com/rfid-starter-kit-for-arduino-uno
- [2] LiquidCrystal I2C library: <https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library>
- [3] Library for the temperature/humidity sensor DHT11: <http://arduino.cc/playground/Main/DHTLib>
- [4] Lib for the RFID-Module: <https://github.com/miguelbalboa/rfid>
- [5] Project page and download pack: www.elektormagazine.com/160322

Listing 2. Display the RFID-UID using the Serial Monitor.

```
// RFID-RC522_data.ino

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

#define RST_PIN      9
#define SS_PIN      10

MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup()
{
  Serial.begin(9600);
  SPI.begin();    // Start SPI bus
  mfrc522.PCD_Init();    // Initialise MFC522 Reader
  mfrc522.PCD_DumpVersionToSerial();    // Show RC522 details
  Serial.println("Place RFID TAG in range!");
}

void loop()
{
  // Card present?
  if ( ! mfrc522.PICC_IsNewCardPresent())
  { return; }

  // Select card
  if ( ! mfrc522.PICC_ReadCardSerial())
  { return; }

  // Send data to serial interface
  mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```