Project: Greenhouse control

By Peer Kröll

Table of contents

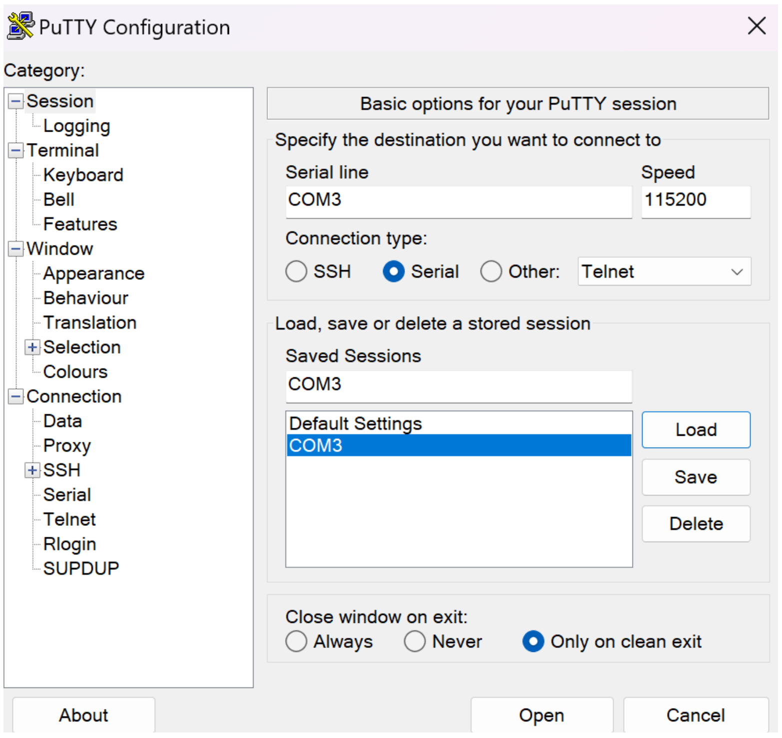
1 User Guide

With this guide the user will be able to get a control system for his greenhouse with the help of a microcontroller and different sensors. You will be able to measure the air humidity, the soil temperature and soil moisture and there is a possibility to open a hatch depending on the temperature and turn on a water sprayer depending on the air humidity, in an automated way. With the provided software and the shown sensors you will get an easy way to create your own greenhouse control system. Before the system works, there are a few steps the user must follow.

First the components have to be wired to specific pins of the microcontroller. There is a list of the components when scrolling down. You will start with the soil temperature sensors, which have to be configurated initially in a specific way. For the Dallas soil sensors to work, their addresses will be needed, because they will be used in a master and slave system over a one wire structure. The wiring gets explained further below in the section wiring.

When the wiring of the Dallas soil sensors is done, the next step is to upload the program to the microcontroller. To do this, you need a program called PSoC Creator, which can be downloaded from the official Creator of Infineon. Furthermore you will also need a program called PuTTY to interact with the program and to print out the outputs and measurements of the microcontroller. When you downloaded the program you have to update the components by clicking on project and update component.

The following picture shows you the PuTTY Configurations. You can find the COM-Port in the device manager.



Specify the destination you want to connect to
Serial line: COM3
Speed: 115200
Connection type: SSH [✓ Serial] Other: Telnet
Load, save or delete a stored session
Saved Sessions: COM3
Default Settings
COM3
Close window on exit: Always Never [✓ Only on clean exit]

Figure 1: PuTTY Configuration

Now you can wire all the other sensors to the microcontroller and when everything is done you are ready to go to use the control system for your greenhouse.

2 Components List

The following components can be used to make the greenhouse control fully functional. In addition the datasheets and suppliers are linked to get more information or to buy the missing parts.

For a small overview, the number of components, the pins and the functions are listed below.

Component	Connected Pin	Function
2 x Dallas DS18B22+/DS1822	P2[6]	Measure soil temperature
1 x Capacitive Soil Moisture Sensor	P0[0]	Measure soil moisture
1 x PWM - Servo Motor	P2[0]	Open and close hatch
1 x DHT11	P1[6]	Measure temperature and air humidity
3 x 4,75kΩ Resistor	-	Pullup resistor
Wires (amount depends on wiring)	-	Connecting/wiring components

Here you'll find a list of the datasheets and suppliers if needed.

Name and datasheet if available	Supplier
Dallas DS18B22+/DS1822	Funduinoshop
Capacitive Soil Moisture Sensor	Mouser
PWM – Servo Motor	Funduinoshop
DHT11	Funduinoshop
Resistor	Reichelt
Jumper wire male/male	Funduinoshop

3 Wiring Diagram

3.1 Components and overview

Before wiring, it is important to know that the red/orange wire is the source (VDD/plus), the black wire is the ground (GND/minus). The green wires give the information/signal in volt that is converted by the AD-converter when measuring the temperature, moisture and air humidity or configuring the servo motor. Additionally, for easier wiring, a breadboard is used and the source and ground are connected to the sideline, making it is easier to connect more components.

The following table provides an overview of the components that are used to make the greenhouse controller work.

Component	Pin
2 x Dallas DS18B22+/DS1822	P2[6]
1 x Capacitive Soil Moisture Sensor	P0[0]
1 x PWM - Servo Motor	P2[0]
1 x DHT11	P1[6]
3 x 4,75kΩ Resistor	-
Wires (amount depends on wiring)	-

3.2 Dallas DS18B22+/DS1822 - One Wire

There are two Dallas DS18B22+ sensors connected in a One Wire system to measure the soil temperature to get two different measurements. Before you can use your sensors, you need to get the addresses of the sensors, which will be explained more precisely now. First, connect only **one** of the two sensors as shown in the picture. But pay attention to the direction of sensor. Here, the green signal wire goes to the pin P2[6], which is in the middle. Between the signal wire and the source (plus) you have to put a pullup resistor (4,75kΩ) for it to work. After running the program, you can press 'o' or 'O' in the user interface. There you have the option to select one of the two memory slots, where you select one for the first sensor and later the other one for the second sensor. After selecting the slot, your address will be stored in the EEPROM of your microcontroller and you can do the same process for the second sensor. So you have to rewire the second one in the same way before you can continue.

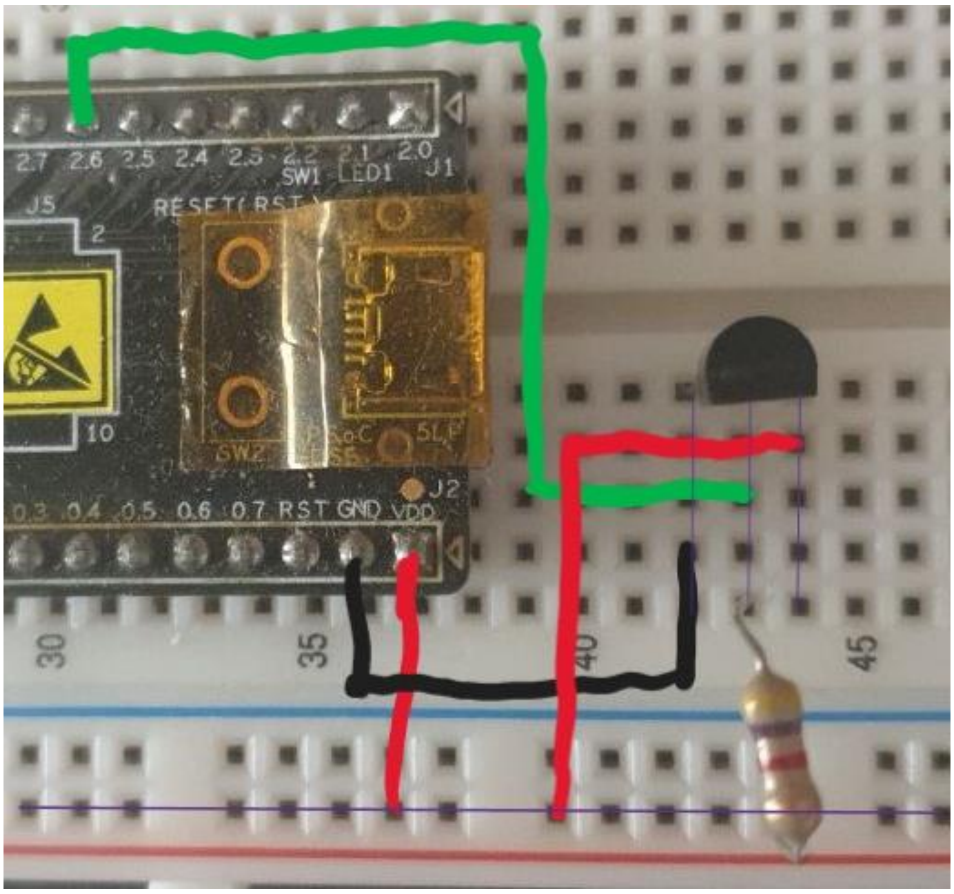


Figure 2: One Dallas sensor

The sideline is used for easier wiring. The thin purple lines show you how the cables are wired underneath and are just for help.

When address saving is done, it is now possible to connect both Dallas sensors to the breadboard as shown. You will need two pullup resistors (4,75kΩ) between the signals and the sources (VDD). Now it should work and the sensors will output both soil temperatures. You can try it out using the live data view option by pressing ‘w’ or ‘W’ in the user menu.

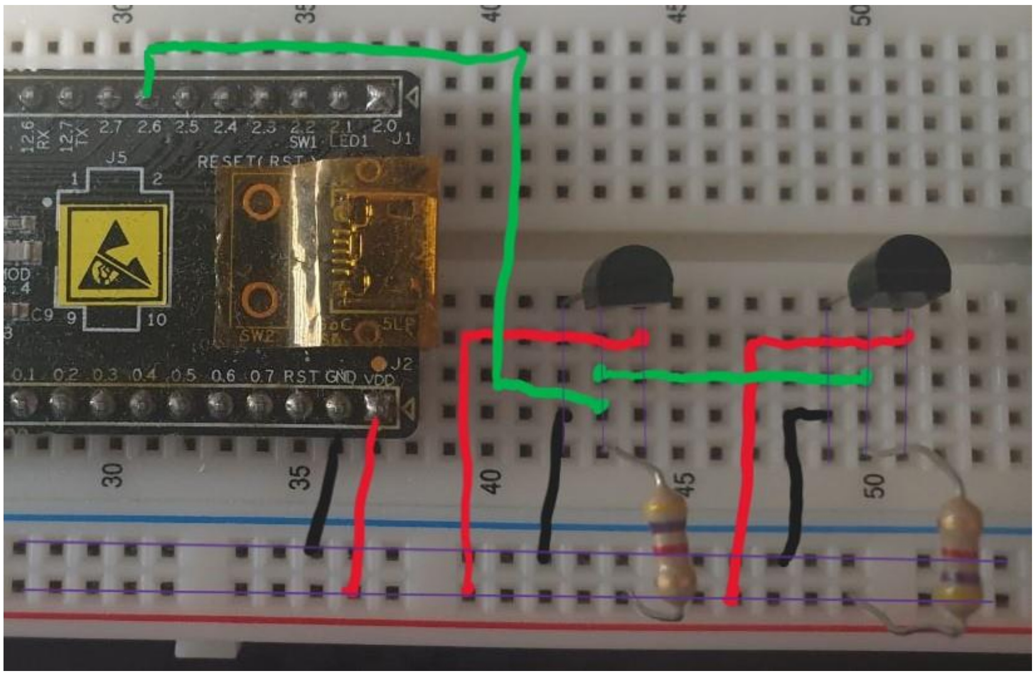


Figure 3: Two Dallas sensors in a One Wire system

3.3 Capacitive Soil Moisture Sensor

The Capacitive Soil Moisture Sensor can be wired as shown in the following picture. The green information wire is connected to the pin P0[0] where the AD conversion takes place. The black wire goes to GND and the red wire goes to VDD as usual.

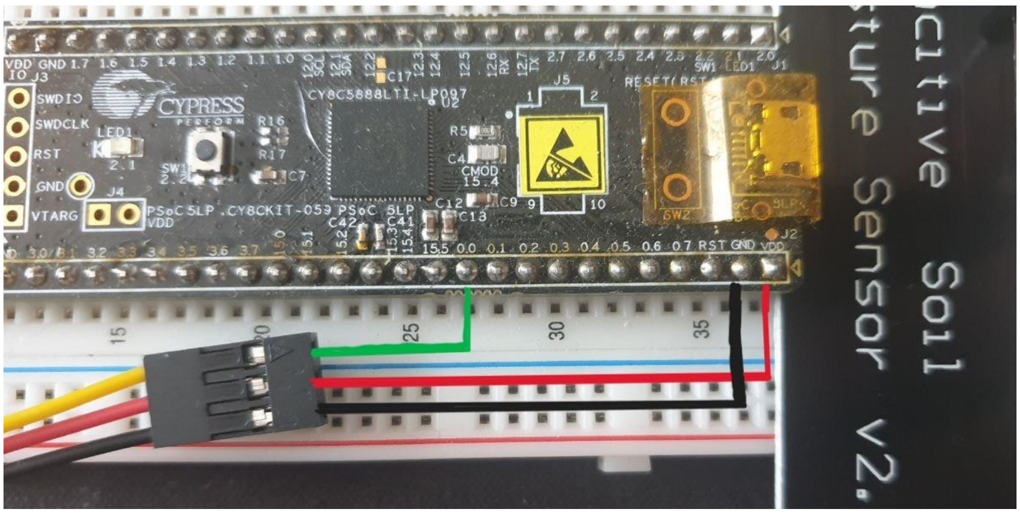


Figure 4: Capacitve Soil Moisture Sensor

3.4 PWM - Servo Motor

As you can see, the PWM - servo motor is connected in the same way as the Capacitive Soil Moisture Sensor, but this time the signal wire is connected to the pin P2[0] this time instead. As mentioned earlier, you can use the side-line which, which is very helpful when you have to wire multiple sensors.

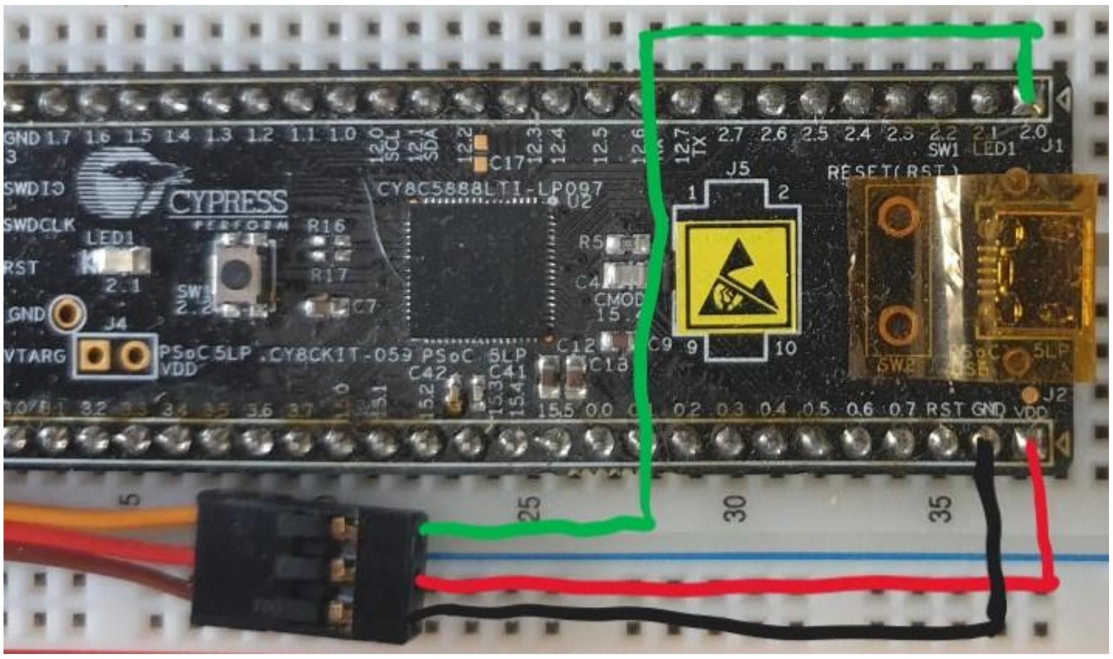


Figure 5: Servo Motor

3.5 DHT11

The next component is the DHT11, which measures the temperature and air humidity. It has three pins, because with one connection the temperature and air humidity are sent over one wire. To make it work you have to connect the VDD (center) and GND (right) as shown below, connect the information wire (left) to pin P1[6] and put a pullup resistor (4,75kΩ) between the VDD and the information wire.

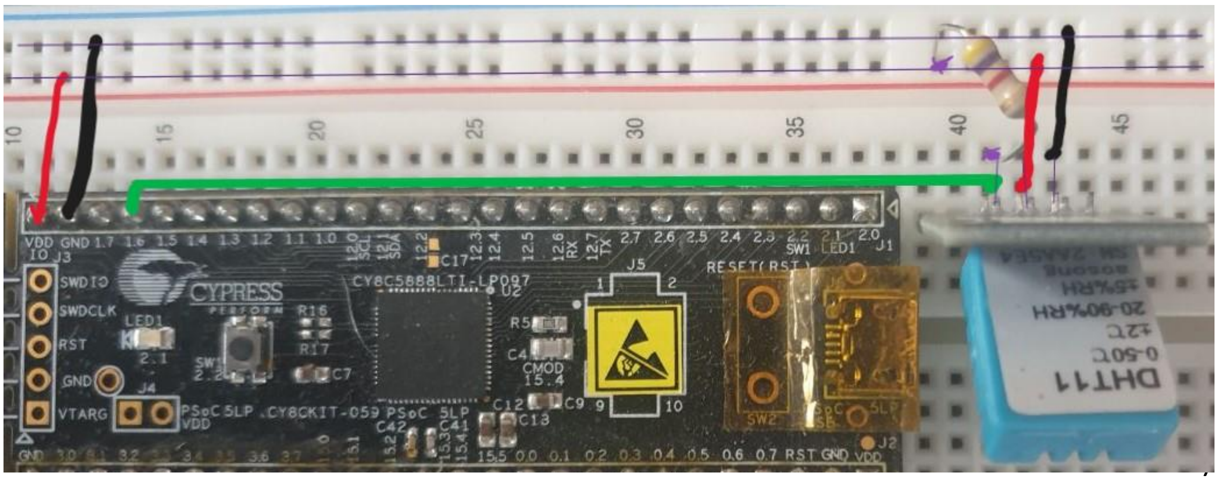
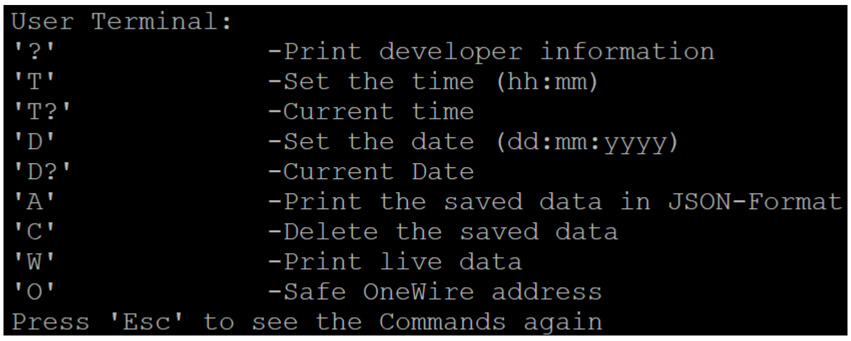


Figure 6: DHT11 sensor

4 User Interface

The user of the program has the possibility to interact with the microcontroller with different inputs by using different keys on the keyboard. When the user uploads the program to the microcontroller and starts the program for the first time, presses the reset button or ‘esc’, the user gets the following output on the console with an overview of the possible interactions with the system:

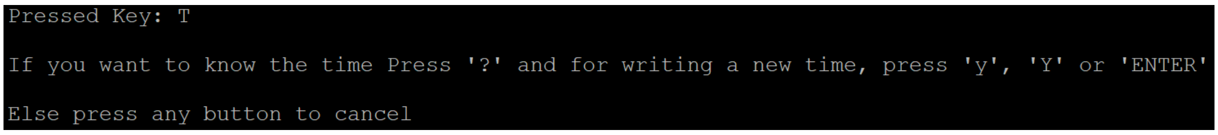


```
User Terminal:
'?'              -Print developer information
'T'              -Set the time (hh:mm)
'T?'             -Current time
'D'              -Set the date (dd:mm:yyyy)
'D?'             -Current Date
'A'              -Print the saved data in JSON-Format
'C'              -Delete the saved data
'W'              -Print live data
'O'              -Safe OneWire address
Press 'Esc' to see the Commands again
```

Figure 7: User command interface

Everything you see above can be written in lower or upper case. If you just press ‘?’, developer information will be printed to the console.

As you can see the user has several options to change the values of the system or just to get information about the measured data. First of all it is possible to set the time and the date of the device or just to print out the current time and date, which has already been stored in the device. Since the microcontroller is not capable to run without a source, the time and date won’t be updated if it is not used. Since the device has no internal clock, the clock had to be programmed separately. Therefore, the user must write the time and date each time the device is disconnected and reconnected to its power source. This is accomplished by pressing ‘T’ or ‘D’ as shown above. While the user is writing the time and date, more help instructions will be seen to make everything easier. Here the time is used as an example.



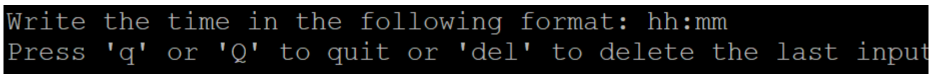
```
Pressed Key: T

If you want to know the time Press '?' and for writing a new time, press 'y', 'Y' or 'ENTER'

Else press any button to cancel
```

Figure 8: Output after pressing 't' or 'T'

If you press '?', the system will print out the saved time, and you can type in the new time you want by pressing 'y', 'Y' or 'ENTER'.



```
Write the time in the following format: hh:mm
Press 'q' or 'Q' to quit or 'del' to delete the last input
```

Figure 9: Writing the time in hh:mm format

Whenever you type something, your input will be printed on the console for a better overview. The same process goes for writing and printing the date as well.

Pressing ‘A’ will print all data already stored in your EEPROM(in JSON format) and to delete all stored data you can press ‘C’. When the EEPROM has been cleared, the user must write a new date and time. To see if all components are working or if there is a problem, the user can print out the live data to the console every few seconds with ‘w’ or ‘W’, which helps to see realistic temperature differences by warming up the sensors, or to see realistic values in general. It than can be canceled by pressing ‘Q’.

The last option is pressing ‘O’ to store the addresses of your Dallas soil temperatures to your EEPROM, which got explained on the chapter ‘’3.2 Dallas DS18B22+/DS1822 - One Wire’’.

5 Bugs and problems

5.1 UART

Sometimes it can happen, that by using the UART for your inputs the buffer/program doesn’t keep up and further inputs cause the program to ‘crashing’ so that it won’t run properly anymore. Often this happens by reconnecting the microcontroller to the source, so you should avoid it. To fix this, you can either restart the program using the restart button or try to press a different button like ‘esc’ and maybe it will work again. Don’t ne surprised if some inputs take slightly longer than expected, because a few commands take time to proceed.

5.2 Timer

While writing the date and time, the program can’t save the data to the EEPROM or update the time. It follows that, when the program wants to save the data at that exact moments, it will need additional seconds/minutes to do so until the user finishes the input. Only then the data gets saved and the time updated. This is because of a loop in the function that firstly doesn’t execute the save data and update time commands and secondly doesn’t leaves until the user is done or cancels the input.