# Control Engineering with Fuzzy Logic

**ektor**books

Practical Applications and Projects with Arduino, ESP32, and RP2040

Fuzzy Triangles Soyo Controller



Josef Bernhardt



### Control Engineering with Fuzzy Logic

## Practical Applications and Projects with Arduino, ESP32, and RP2040

Josef Bernhardt



 This is an Elektor Publication. Elektor is the media brand of Elektor International Media B.V.
 PO Box 11, NL-6114-ZG Susteren, The Netherlands
 Phone: +31 46 4389444

• All rights reserved. No part of this book may be reproduced in any material form, including photocopying, or storing in any medium by electronic means and whether or not transiently or incidentally to some other use of this publication, without the written permission of the copyright holder except in accordance with the provisions of the Copyright Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licencing Agency Ltd., 90 Tottenham Court Road, London, England W1P 9HE. Applications for the copyright holder's permission to reproduce any part of the publication should be addressed to the publishers.

#### Declaration

The author and publisher have made every effort to ensure the accuracy of the information contained in this book. They do not assume, or hereby disclaim, any liability to any party for any loss or damage caused by errors or omissions in this book, whether such errors or omissions result from negligence, accident, or any other cause.

### ISBN 978-3-89576-660-2 Print ISBN 978-3-89576-661-9 eBook

 © Copyright 2025 Elektor International Media www.elektor.com
 Editor: Ferdinand te Walvaart
 Prepress Production: D-Vision, Julian van den Berg
 Printers: Ipskamp, Enschede, The Netherlands

Elektor is the world's leading source of essential technical information and electronics products for pro engineers, electronics designers, and the companies seeking to engage them. Each day, our international team develops and delivers high-quality content - via a variety of media channels (including magazines, video, digital media, and social media) in several languages - relating to electronics design and DIY electronics. **www.elektormagazine.com** 

#### Contents

| Pref | ace  |
|------|--|
| Cha  | pter 1 • Arduino Hardware for Control Engineering      |
| 1    | .1 Arduino UNO   |
| 1    | 2 Arduino NANO   |
| 1    | 3 Espressif ESP32                                      |
| 1    | .4 Raspberry RP2040 controller                         |
| 1    | 5 ESP32 Fuzzy PID controller board15                   |
| 1    | 6 ESP32 Modbus RTU controller board19                  |
| 1    | 7 Programming the Arduino boards                       |
| 1    | 8 The Nextion graphic display                          |
| 1    | .9 Test software for the ESP32 board                   |
| 1    | .10 Program example with the ESP3228                   |
| 1    | .11 Arduino ESP32 controller 12 bit A/D converter test |
| 1    | .12 CodeBlocks development environment for Ansi C      |
| 1    | .13 Serial Data Reader                                 |
| 1    | .14 Serial Port Data Reader                            |
| 1    | .15 The Espressif ESP32 as a WebSocket server          |
| Cha  | pter 2 • Sensors for Control Engineering54             |
| 2    | 2.1 NTC temperature sensors                            |
| 2    | 2.2 RTD temperature sensors                            |
| 2    | 2.3 Linear Interpolation                               |
| 2    | 2.4 Thermocouples                                      |
| 2    | 2.5 Applications of thermocouples                      |
| 2    | 2.6 Sensors with integrated electronics                |
| 2    | 2.7 pH sensors   |
| 2    | 2.8 Rotary encoder                                     |
| 2    | 2.9 Force measurement                                  |
| 2    | 2.10 Linear regression                                 |
| 2    | 2.11 Spline interpolation                              |
| 2    | 2.12 Polynomial regression                             |

| Cha | apter 3 • Actuators - MOSFET and SSR87  |
|-----|---|
|     | 3.1 MOSFET BSP76  |
|     | 3.2 MOSFET circuit with optocoupler for 10 A  |
|     | 3.3 H-bridge L298   |
|     | 3.4 H-bridge with BTS7960 from Infineon up to a maximum of 43 A                       |
|     | 3.5 Thyristor actuator for AC voltage   |
| Cha | apter 4 • Control, regulation and control systems                                     |
|     | 4.1 Formula symbols   |
|     | 4.2 Difference between control and control with feedback according to IEC 60050-35192 |
|     | 4.3 Examples control technology   |
|     | 4.4 Control of a real heating system  |
|     | 4.5 Control of a DC motor with PWM  |
| Cha | apter 5 • On-Off and PID controllers  |
|     | 5.1 Two-point controller  |
|     | 5.2 PID controller  |
|     | 5.3 Parameter determination using Ziegler-Nichols                                     |
|     | 5.4 PID parameter determination with the relay method                                 |
|     | 5.5 Arduino PID controller with PT2 simulation  |
|     | 5.6 Arduino PID controller with real controlled system (ALU block)                    |
|     | 5.7 Arduino PID controller for standing pendulum                                      |
| Cha | apter 6 • Fuzzy Logic basics  |
|     | 6.1 Fuzzy Logic history   |
|     | 6.2 Fuzzy controller structure  |
|     | 6.3 Fuzzification   |
|     | 6.4 Inference   |
|     | 6.5 Defuzzification   |
|     | 6.6 Fuzzy controller example for a fan  |
|     | 6.7 Control of an inverted pendulum   |
|     | 6.8 Fuzzy sets  |
|     | 6.9 Simulation of a function with fuzzy logic   |
|     | 6.10 Application of a fuzzy function for a temperature profile                        |

| Cha | apter 7 • Fuzzy Logic temperature controller   | 86 |
|-----|--|----|
|     | 7.1 Fuzzy controller with PT2 simulation1  | 86 |
|     | 7.2 Fuzzy controller with the ESP32 and real heating   | 93 |
|     | 7.3 Fuzzy examples with Gauss, Bell and Sigmoid  | 06 |
| Cha | apter 8 $\bullet$ Fuzzy logic temperature controller with two inputs2                                | 07 |
|     | 8.1 Temperature controller with two inputs2  | 07 |
|     | 8.2 Fuzzy control example for a heating control system   | 11 |
| Cha | apter 9 • Fuzzy controller examples with a DC motor  | 20 |
|     | 9.1 DC motor basics  | 20 |
|     | 9.2 Control of the motor with PWM (speed control)  | 22 |
|     | 9.3 Motor control with PWM and speed measurement   | 23 |
|     | 9.4 Motor speed control with fuzzy logic2  | 27 |
|     | 9.5 Motor position control   | 40 |
| Cha | apter 10 • Sliding mode controller2  | 50 |
|     | 10.1 General description of the sliding mode controller  | 50 |
|     | 10.3 Sliding mode controller example with real system  | 56 |
|     | 10.4 Optimization of the parameters $c_1$ and $c_2$ with fuzzy logic $\ldots \ldots \ldots \ldots 2$ | 68 |
| Cha | apter 11 • Neural networks   | 72 |
|     | 11.1 History of neural networks2   | 72 |
|     | 11.2 Biological neuron   | 73 |
|     | 11.3 Basics  | 73 |
|     | 11.4 Activation functions  | 74 |
|     | 11.5 The perceptron  | 76 |
|     | 11.6 Feedforward networks2   | 76 |
|     | 11.7 Training the neural network2  | 79 |
|     | 11.8 ANN example XOR   | 82 |
|     | 11.9 Fuzzy Gauss controller with neural network  | 87 |
|     | 11.10 Color sensor evaluation with neural network  | 94 |
|     | 11.11 Training a neural network with two hidden layers   | 07 |
| Cha | apter 12 $ullet$ Fuzzy controller for feeding electricity into the domestic grid3                    | 20 |
|     | 12.1 Introduction  | 20 |
|     | 12.2 Shelly 3EM Energy Meter   | 20 |

| Index | <i>ι</i>   | .356  |  |  |  |  |  |
|-------|--|-------|--|--|--|--|--|
| Apper | nix C • Weblinks   | .354  |  |  |  |  |  |
| Apper | ndix B • Bibliography                                    | .353  |  |  |  |  |  |
| Apper | Appendix A • Circuits layouts and parts list             |       |  |  |  |  |  |
| 12.   | .8 Fuzzy controller function                             | . 336 |  |  |  |  |  |
| 12.   | .7 ESP32 PID Fuzzy Control Board for closed-loop control | . 331 |  |  |  |  |  |
| 12.   | .6 Soyo Micro Inverter for feeding into the grid         | . 329 |  |  |  |  |  |
| 12.   | .5 Windows software SharpDevelop_Shelly3emWinform        | . 329 |  |  |  |  |  |
| 12.   | .4 Charge controller - Victron Smart Solar 100 I 20      | . 326 |  |  |  |  |  |
| 12.   | .3 Measurements with the Shelly Plus 1PM                 | . 323 |  |  |  |  |  |

#### Preface

This book is written for students and electronics enthusiasts interested in practical applications of control engineering. The aim of this book is to teach the fundamentals of control engineering, with a special focus on Fuzzy Logic control systems, and to illustrate these concepts through various practical examples.

In particular, the basics of sensors and linearization are presented in an easily understandable manner, providing a solid foundation for acquiring sensor data. Furthermore, we offer detailed information and examples on controlling different actuators, demonstrating how you can build your own projects.

The book is supplemented with numerous examples and describes real-world applications of control engineering, all implemented on the Arduino platform. This approach allows readers to gain hands-on experience and apply what they have learned directly in practice.

In addition to covering the basics of the Arduino Uno, we also delve into the use of the powerful ESP32 to give you a comprehensive insight into the world of microcontrollers. You will be guided step-by-step through applications for temperature control using Fuzzy Logic, and learn how to control motors, both in terms of speed and position.

The book concludes with chapters on sliding mode controllers, the basics of neural networks, and controllers for grid feed-in, all complemented with practical examples.

We hope this book not only provides enjoyment while reading but also assists you in successfully implementing your own projects. We wish you much fun and success with this book and your future projects!

Finally, we would like to thank everyone who supported us, repeatedly checked details, read the manuscript, provided feedback, allowed us to quote their comments, and helped with editing, proofreading, and design.

Josef Bernhardt

#### **Chapter 1 • Arduino Hardware for Control Engineering**

Arduino is an open-source platform for electronics projects that combines hardware and software. It is user-friendly and suitable for both beginners and experienced developers. There is a wide variety of Arduino boards, sensors, and components to choose from.

The name Arduino originates from a bar in the town of Ivrea, Italy, where in 2005, students Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis conceived the idea of developing a programmable board for prototyping. Arduino was quickly released as an open-source platform, which led to the founding of the Arduino company in 2009. The company supports the production and distribution of the boards and continually develops new products.

With Arduino boards, projects can be realized in various fields, such as robotics and industrial control systems. For programming the hardware platform, the Arduino IDE is used, which allows users to write, test, and upload programs.

The IDE is based on C/C++ and makes it easier for beginners to get started by offering numerous libraries and example codes. Arduino provides endless possibilities for creativity and innovation.

In this book on control engineering, the focus is mainly on the Arduino UNO, Arduino Nano, and the ESP32 from Espressif. The Raspberry RP2040 is also of great interest for more complex projects.

#### **1.1 Arduino UNO**

The Arduino UNO is one of the most popular Arduino boards and is based on the ATmega328P microcontroller. It offers numerous digital and analog pins, as well as features like analog/ digital converters, PWM outputs, and serial communication. The UNO is easy to use and has a large developer community, making it ideal for control engineering. Programming is done via the Arduino IDE using C code. With the UNO, we can control sensors, heaters, motors, LEDs, and other components, and build high-quality controllers. Its functionality can be extended with expansion boards, such as displays, network connections, or wireless communication. The UNO is widely used in educational and hobby projects, as well as in professional applications for prototyping and rapid development.



Figure 1.1: Arduino UNO Board

#### **1.2 Arduino NANO**

The Arduino Nano is highly suitable for electronics projects due to its compact form factor and is also based on the ATmega328P microcontroller. It offers similar features to the Arduino UNO but in a smaller size, making it ideal for space-constrained projects. The Nano has 14 digital I/O pins, 8 analog inputs, and supports serial communication as well as PWM outputs. Due to its small size and flexibility, the Arduino Nano is often used in portable and space-critical applications.



Figure 1.2: Arduino Nano board

#### Features and Specifications of the CPU

- Microcontroller: Atmel / Microchip ATmega328P
- Operating Voltage: 5 Volts
- Input Voltage (recommended): 7-12 Volts
- Digital I/O Pins: 14 (6 of which support PWM)
- Analog Input Pins: 6 with 10-bit resolution
- Flash Memory: 32 KB (0.5 KB reserved for the bootloader)
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Frequency: 16 MHz

| Arduino Nano Pin | Board     | Description               |
|------------------|-----------|---------------------------|
| D2               | Button S1 | Digital Input active High |
| D3               | Button S2 | Digital Input active High |
| D4               | Button S3 | Digital Input active High |
| D5               | Button S4 | Digital Input active High |
| D6               | Button S5 | Digital Input active High |
| D7               | LED 1     | Digital Output            |
| D8               | LED 2     | Digital Output            |
| D9               | LED 6     | PWM Output                |
| D10              | LED 7     | PWM Output                |
| D11              | LED 8     | PWM Output                |
| D12              | LED 3     | Digital Output            |
| D 13             | LED 4     | Digital Output            |
| ADC0             | Poti R13  | Analog Input              |
| ADC1             | Poti R14  | Analog Input              |
| ADC2             | Poti R15  | Analog Input              |
| ADC3             | Poti R16  | Analog Input              |
| ADC4             | Poti R17  | Analog Input              |

Table 1.1: Pin Assignment Arduino Nano test board.



Figure 1.3: Arduino Nano Test board

#### 1.3 Espressif ESP32

The ESP32 from Espressif is a powerful development board based on the ESP32 microcontroller. It features integrated Wi-Fi and Bluetooth, making it ideal for IoT applications. Programming is also done through the Arduino IDE, which can be downloaded from the official Arduino website [LINK1]. After installing the ESP32 board support, we can program the ESP32 using C++ code and access its hardware. The ESP32 has numerous GPIO pins for connecting sensors and actuators, and it can function as both a server and a client in Wi-Fi networks. Additionally, it allows communication with other devices via Bluetooth.

Due to its versatility, the ESP32 is suitable for projects in smart home, robotics, and control engineering. It is also beginner-friendly and benefits from a large developer community and numerous online resources that help with getting started. With the ESP32, we can collect sensor data and display it over Wi-Fi or on a TFT display, as well as control actuators.



Figure 1.4: ESP32 board

#### Technical Specifications of the ESP32 Microcontroller:

- Processor: Dual-Core Tensilica LX6 with up to 240 Mhz
- Memory:
  - RAM: 520 KB SRAM
  - **ROM**: 448 KB
  - Flash Memory: Varies by model, typically 4 MB

#### • Features:

- Wi-Fi: 802.11 b/g/n
- Bluetooth: v4.2 BR/EDR and BLE
- GPIO Pins: 34 pins, versatile for various peripherals

#### • Communication Interfaces:

- UART: 3 interfaces
- SPI: 4 interfaces
- I<sup>2</sup>C: 2 interfaces
- I<sup>2</sup>S: 2 interfaces
- PWM: Up to 16 channels
- DAC: 2 channels
- ADC: Up to 18 channels, 12-bit resolution

#### • Additional Features:

- Touch Sensors: Up to 10 capacitive touch inputs
- RTC (Real-Time Clock)
- Hardware Acceleration for Encryption: AES, SHA-2, RSA, ECC

#### • Power Management:

- Various power-saving modes (Light Sleep, Deep Sleep)
- Operating Voltage: 2.2 V 3.6 V
- Operating Temperature: -40°C to +85°C
- Peripherals:
  - Hall Sensor
  - Temperature Sensor
- Connectivity: USB, external power supply via battery or rechargeable battery

#### 1.4 Raspberry RP2040 controller

The RP2040 is a powerful microcontroller developed by the Raspberry Pi Foundation. It is an affordable microcontroller based on a dual-core architecture, specifically designed for embedded applications. It stands out for its high performance, flexibility, and energy efficiency.

The CPU features a range of functions, including GPIO pins, UART, SPI, and I<sup>2</sup>C interfaces, as well as a variety of peripheral modules. It also supports programming via USB and includes an integrated USB bootloader, enabling programming and operation without the need for external programming devices.



Figure 1.5: Raspberry RP2040 Board

#### **Features and Specifications:**

- Microcontroller: Dual-Core ARM Cortex-M0+ (up to 133 MHz)
- Operating Voltage: 3.3 Volts
- **Digital I/O Pins**: 30 (26 of which are PWM-capable)
- Analog Input Pins: 4
- Flash Memory: 2 MB (QSPI)
- **SRAM**: 264 KB
- **EEPROM**: Not available
- Clock Speed: Up to 133 MHz

The RP2040 is supported by a dedicated developer community that provides a variety of resources, including documentation, tutorials, forums, and example projects. These resources make project development easier and encourage the exchange of knowledge and experiences within the community.

#### **1.5 ESP32 Fuzzy PID controller board**

For the program examples in this book, we use a test board specifically designed for control engineering. The "ESP32 Fuzzy PID Controller" board is a versatile platform for precise control and regulation applications. With a wide range of interfaces, it allows seamless integration into numerous applications.

Additional practical applications, such as using it as a PLC or Modbus TCP I/O module, as well as protocols like MQTT or web applications with WebSockets, are always possible.



Figure 1.6: ESP32 Fuzzy PID Test board

#### Input Interfaces:

- **2 x NTC 10K**: These interfaces enable precise temperature change measurements using NTC temperature sensors.
- **1 x PT1000**: The PT1000 interface allows for accurate temperature readings using PT1000 temperature sensors.
- 1 x OneWire for DS18B20 or DHT11 / DHT22: This flexible interface supports the use of DS18B20 temperature sensors or DHT11/DHT22 humidity sensors for reliable environmental monitoring.
- 2 **x Buttons**: Two buttons provide an easy user interface for controlling and configuring the system.
- **2x Potentiometers**: Two potentiometers allow manual adjustment of parameters or thresholds for fine-tuned control.
- 2 x Sense Lines for Motor Speed and Position Monitoring: These interfaces allow for precise measurement of motor speed and position for accurate control.

#### **Output Interfaces:**

• 2x 24V 1A for Motor or Relays, optionally also for PWM: These outputs offer flexibility for controlling motors or relays, optionally via PWM signals for precise power control.

- 2 x Analog Outputs 0 to 5 Volts: Two analog outputs enable the generation of voltage signals in the range of 0 to 5 volts for driving external devices.
- **1 x 4-pin I2C Connection for Expansion**: The I2C interface allows easy integration of expansion modules for additional functions or interfaces.
- 1 x UART2 TTL for Nextion Display NX3224F028 (2.8 inches): The Nextion display provides a clear, user-friendly interface for displaying data and, with touch functionality, allows interaction with the system.

With its powerful ESP32 chip and a wide range of interfaces, the ESP32 Fuzzy PID Controller Board offers a robust and flexible solution for a variety of applications, from temperature control to position monitoring. A complete layout and schematic diagrams can be found in the appendix.

#### Pin Assignment of the ESP32 from the Test Board:



Figure 1.7: ESP32 Pinout

| Pin Description | ESP32 Modul Pin Nr. | GPIO pin<br>number | Description                          |
|-----------------|---------------------|--------------------|--------------------------------------|
| VCC             | 30                  |                    | VCC Pin 5V from USB                  |
| 3V3             | 1                   |                    | 3.3 Volt pin from voltage regulator  |
| GND             | 2 / 29              |                    | GND Pin                              |
| ADC1            | 21                  | IO32               | Analogue digital converter channel 1 |
| ADC2            | 22                  | IO33               | Analogue digital converter channel 2 |
| BUTTON1         | 26                  | IO14               | Button 1 Input                       |

| BUTTON2  | 25 | IO27 | Button 2 Input          |
|----------|----|------|-------------------------|
| MOTSENS1 | 3  | IO15 | Motor Sensor 1 Input    |
| MOTSENS2 | 19 | IO34 | Motor Sensor 2 Input    |
| I2CSDA   | 11 | IO21 | I <sup>2</sup> C Data   |
| I2CSCL   | 14 | IO22 | I <sup>2</sup> C Clock  |
| RXD2     | 6  | IO16 | UART RX Input           |
| TXD2     | 7  | IO17 | UART TX Output          |
| DS18B20  | 5  | IO4  | OneWire Interface       |
| DAC1     | 23 | IO25 | Analog Output 1         |
|          |    |      | 0 to 5.0 V              |
| DAC2     | 24 | IO26 | Analog Output 2         |
|          |    |      | 0 to 5.0 V              |
| PWM1     | 27 | IO12 | PWM Output 1            |
| PWM2     | 28 | IO13 | PWM Output 2            |
| RELAIS1  | 9  | IO18 | Activation for relays 1 |
| RELAIS2  | 10 | IO19 | Activation for relays 2 |
| LED1     | 8  | IO5  | LED Board               |

Table 1.2: Pin assignment for the GPIO ESP32 module



Figure 1.8: ESP32 Fuzzy PID Board

#### 1.6 ESP32 Modbus RTU controller board

This board is also suitable for capturing measurement values such as current, voltage, and power, and is known as the **ESP32BoardModbusRTUControl** board. Schematic, diagrams and layouts are included in the appendix.

The ESP32 board is designed to read energy meters such as the Shelly 3EM via Wi-Fi, as well as the Eastron SDM series (compatible with Controlin) and KBR multimess 96 via RS485 (Modbus RTU). A functioning Wi-Fi connection is required for operation. The data from the meters is displayed on the built-in screen and sent to a server every second in JSON format via HTTP, where it is stored. XML format is also possible. The controller software was developed using the Arduino IDE. Possible protocols include HTTP POST and GET, MQTT, REST, Modbus TCP Server, and other TCP/IP protocols.

The board also allows for the control of inverters to achieve zero injection using the power data.

#### **Technical Specifications:**

- CPU: ESP32 by Tensilica
- Power Supply: 12-24 Volts DC
- Interfaces:
  - RS485
  - Wi-Fi 2.4 GHz and Bluetooth
  - UART RX TX for 5V Nextion Display 2.4-inch NX3224T024

#### • Outputs:

- 2 x 24V DC outputs for relays or contactors

#### • Inputs/Outputs:

- 1 x 10V analog input (expandable to 0-20V or 0-50V)
- 1 x 10V analog output (or 0-5V)
- 1 x OneWire for DS18B20 temperature sensor

#### Information on the Software of the ESP32 Module:

The module automatically connects to the Wi-Fi network after power-on, using the SSID and password that are stored in the program. If no Wi-Fi network is available, an access point must be installed.

#### Listing: ESP32\_SDM\_EnergyMeter.ino

```
// Wlan Parameter
char* ssid = "FRITZ!Box 7590 RI";
char* password = "52x6165x80538x77x967";
```

You then have the option of either reading the Shelly 3EM via WLAN or the Eastron SDM or Controlin meter via Modbus RTU.

Here is an example of querying the Shelly 3EM every second:

```
// Shelly 3 em
String serverName1 = "http://192.168.178.40/emeter/0";
/// Answer from Shelly
{"power":15.32,"pf":0.25,"current":0.27,"voltage":226.96,"is_
valid":true,"total":2.0,"total_returned":0.0}
String serverName2 = "http://192.168.178.40/emeter/1";
// Answer from Shelly
{"power":49.16,"pf":0.78,"current":0.27,"voltage":225.98,"is_
valid":true,"total":5.9,"total_returned":0.0}
String serverName3 = "http://192.168.178.40/emeter/2";
/// Answer from Shelly
{"power":11.56,"pf":0.40,"current":0.13,"voltage":227.10,"is_
valid":true,"total":2.0,"total_returned":0.0}
```

The data is converted from JSON format to float values, and the total performance is calculated.

The current timestamp is then queried from a time server and the data is sent to the Firebase or Postgre SQL Server.

An RS485 energy meter is connected via the UART1 and the following library:

https://github.com/reaper7/SDM\_Energy\_Meter

```
// Nextion Display
#define RXD2 16
#define TXD2 17
// RS485 Modbus RTU
#define RXD1 19
#define RXD1 18
#define REDE1 26
// Init RS485 Modbus RTU
SDM sdm(Serial1, SDM_UART_BAUD, REDE1, SERIAL_8N1, RXD1, TXD1);
```

The following Modbus registers are read out:

```
PowerArr[0] = sdm.readVal(SDM_PHASE_1_VOLTAGE);
PowerArr[1] = sdm.readVal(SDM_PHASE_1_CURRENT);
PowerArr[2] = sdm.readVal(SDM_PHASE_1_POWER);
PowerArr[3] = sdm.readVal(SDM_FREQUENCY);
PowerArr[4] = sdm.readVal(SDM_IMPORT_ACTIVE_ENERGY);
PowerArr[5] = sdm.readVal(SDM_EXPORT_ACTIVE_ENERGY);
```

For Modbus RTU meters from other manufacturers, the registers must be adapted and tests carried out.



Figure 1.9: ESP32 Modbus RTU Board

The appendix also contains circuit diagrams and layouts for an Arduino Nano board and an Arduino Mega board with four digital and four analogue lines.

#### **1.7 Programming the Arduino boards**

A development environment is available for programming the boards, which can be downloaded for free from the Arduino website.

The software is available for Windows, Linux, and macOS.

The website address is: https://www.arduino.cc/en/software



Figure 1.10: Download Arduino IDE

After installing the IDE, missing boards such as the ESP32 can be installed through the Board Manager settings. A list of installable Arduino boards can be found here:



Figure 1.11: ESP32 Github website

Unofficial list of 3rd party boards support urls  $\cdot$  arduino/Arduino Wiki  $\cdot$  GitHub

- 1. Open the Arduino IDE 2.3.2 on your computer.
- 2. Go to 'Tools' > 'Board' > 'Board Manager'.

|   | Einstell        | ungen  | Netzwerk         |             |
|---|-----------------|--------|------------------|-------------|
| Dateipfad des Sketchbooks:  |                 |        |                  |             |
| c:\Users\LocalAdmin\Documents\Ardu  | iino            |        |                  | DURCHSUCHEN |
| Dateien im Sketch zeigen  |                 |        |                  |             |
| Editor Schriftgröße:  | 14              |        |                  |             |
| Größe der Benutzeroberfläche:   | Automatisch     | 100    | %                |             |
| Farbdesign:   | Hell            | ,      | ~                |             |
| Editorsprache:  | Deutsch         | ✓ (R   | teload required) |             |
| Compiler-Meldungen anzeigen beim  | 🗌 Kompilieren 🗌 | Hochla | aden             |             |
| Compiler-Meldungen  | Kein/e/r 🖌      |        |                  |             |
| Code nach Hochladen überprüfen<br>Automatisch speichern<br>Schnelle Editor Vorschläge |                 |        |                  |             |
| Zusätzliche Boardverwalter-URLs:  |                 |        |                  | 6           |
|   |                 |        |                  |             |
|   |                 |        |                  |             |

Figure 1.12: Settings in the Arduino IDE

- 3. Enter 'ESP32' in the search bar at the top right of the board manager and press Enter.
- 4. A list of available ESP32 boards should appear. Select the ESP32 board you want to use (e.g., 'ESP32 Dev Module').
- 5. Click on the Install button next to the ESP32 board to install the board support.
- 6. Wait until the installation is complete. This will take a few minutes.
- 7. Once the installation is complete, you can select the ESP32 board from the list of available boards under 'Tools' > 'Board'.

After the ESP32 board is successfully installed, you can use it like any other board in the Arduino IDE and write and upload programs for the ESP32.

#### **1.8 The Nextion graphic display**

To display the measurement data in the examples, we use a 2.8 inch graphic display NX3224F028\_001 from Nextion. The big advantage over other displays, such as those with an IL9341 graphics chip, is the control with the UART via the RX/TX lines and, above all, the simple programming.

A free development environment for the PC is available from Nextion for programming the display. This can be used to design the user interface for the display on the PC. Many graphic elements such as buttons, text boxes, labels etc. are available.

| 🕼 Netrion Editor/C/U.Bert/LocalAdmin/Documents/Betor/Fizzy_Logik_Buch/Netrion/ControllerDirplay/HM) =                                       |           |  |   |                        |      |           |   | a x |            |             |   |
|---|-----------|--|---|------------------------|------|-----------|---|-----|------------|-------------|---|
| File Tools Setting Help About Schließen   |           |  |   |                        |      |           |   |     |            |             |   |
| 🚰 Coen 🗋 New 💾 Save 🕮 Coencile 🕘 Debua 🖡 Uplost 🔚 Copy 🖧 Cut 📲 Paste - 🚔 Lock 着 Unicat 🗙 Delete 🔊 Unicol). 🕐 Redo(0) 🚳 Device ID 150% 💬 🕢 🛞 |           |  |   |                        |      |           |   |     |            |             |   |
| ↑ 上 国 司 山 43 A 田 10+ 00+ 00 名 名 名 さ! C // ! //.   |           |  |   |                        |      |           |   |     |            |             |   |
| Toolhox   | 4         |  |   |                        |      |           |   | _   | Page       |             |   |
| A Text  |           | Program.s ^  |   |                        |      |           | - | -   |            |             | - |
| A Scrolling text  |           |  | ‴ Fl  | JZZY PIE               | C    | ontroller |   | Î   |            |             | _ |
| a 123 Number  |           |  | -   |                        | _    |           |   |     | o a pageo  |             |   |
| 2 1. Xfloat   |           |  | " т   | 1                      |      | 0.0       |   |     |            |             |   |
| Button  |           |  | 12  |                        | -    |           |   |     |            |             |   |
| Progress bar  |           |  | " т   | 2                      | 0012 | 0.0       |   |     |            |             |   |
| Picture   |           |  | 13  |                        | INT3 |           |   |     |            |             |   |
| ⊁ Crop  |           |  | - т   | 3                      |      | 0.0       |   |     |            |             |   |
| Touch Car   |           |  | 14  |                        | bdT4 | 0.0       |   |     |            |             |   |
| Gauge   |           |  |   | 4                      |      | 0.0       |   |     | Attribute  |             | 4 |
| √ Waveform  |           |  | 15 -  | -                      | bdT5 | 0.0       |   |     | page0(Page | e)          |   |
| - Slider  |           |  |   | 5                      |      | 0.0       |   |     | 😤 🛶        |             |   |
| () Timer  |           |  | <sup>16</sup> T                                       | 6                      | bdT6 | 0.0       |   |     | type       | 121         |   |
| (X) Variable  |           |  |   | 0                      |      | 0.0       |   |     | id         | 0           |   |
| Dual-state button   |           | 4  | II  |                        |      |           |   |     | vscope     | local       |   |
| Checkbox  |           | Event  | ent P Outert  |                        |      |           |   |     | sta        | solid color |   |
| Radio     Redio   |           | Preinitialize. Postinitializ.                      | Touch Pres_ Touc                                      | h Rele., Page Exit E., |      |           |   |     | bco        | 10597       |   |
| 28 GRcode   |           | //= //¥  |   |                        |      |           |   |     | x          | 0           |   |
| Switch  |           | ( Perinitalian super encode                        |   |                        |      |           |   |     | w          | 320         |   |
| TextSelect  |           | (Freminalize evenicexecute                         | "reinitialize event execute before component refresh) |                        |      |           |   |     | h          | 240         |   |
| TI SLText   |           |  |   |                        |      |           |   |     |            |             |   |
| DataRecord  |           |  |   |                        |      |           |   |     |            |             |   |
| FileBrowser   |           |  |   |                        |      |           |   |     |            |             |   |
| FileStream  |           |  |   |                        | -    |           |   |     |            |             |   |
| Gmov  |           |  |   |                        |      |           |   |     |            |             |   |
| UVideo  |           | Click the attribute to display corresponding notes |   |                        |      |           |   | .es |            |             |   |
|   |           |  |   |                        |      |           |   |     |            |             |   |
| Encoding iso-8859-1   Model:NX3224F028_011 inch:2.8(2-  | 40X320) F | lash:4M RAM:3584B Frequenc                         | y64M  Coordinate>                                     | (442 Y:186             |      |           |   |     |            |             |   |

Figure 1.13: Nextion Editor IDE

Also included in the IDE is a debug function and an option to transfer the TFT file to the display. A simple USB-UART module can be used for this (upload).

A second option is to save the generated TFT file to a micro-SD card. This menu item can be found under 'File' 'TFT file output.' The SD card is then inserted into the SD slot of the display and voltage is applied. The program is now transferred from the display to the microcontroller. The card is then removed again. This saves you having to plug and unplug the display when it is mounted on the ESP32 board.

The ControllerDisplay.HMI and ControllerDisplay.tft files are stored in the 'Nextion' directory.

You can find lots more information and tutorials on the Nextion website.

#### 1.9 Test software for the ESP32 board

This program is for initializing the hardware. The timer is initialized to 10 ms using the Ticker.h library. Then the two UART for the USB connection to the PC and UART2 for the Nextion display. In the timer the variable counter10ms is increased, if the value 100 is reached it sets a flag oneSecond which is queried in the loop(). This provides an exact time cycle for the output of the second counter. The program can be used as a 'basic framework' for further programs.

#### Listing 1: Arduino\_UART2\_NextionDisplay.ino

```
//--
11
11
             Arduino Testsoftware
             _____
11
11
       Testprogramm for Arduino ESP32 PID Fuzzy Control Board
11
       10ms Timer Interrupt Function
11
       Init UART2 for Nextion Display
11
11
       Display seconds on Display NX3224F028_001
11
// Date: 28/04/2024
// Author: Josef Bernhardt
// Hardware: Arduino (ESPRESSIF) ESP32
11
// File:
            Arduino UART2 NextionDisplay.ino
//-----
#include <Arduino.h>
#include <Ticker.h>
int LED_BUILTIN = 2;
// ESP32 UART 2 Pins
#define RXD2 16
#define TXD2 17
// I2C Pins
#define I2CSCL 22
#define I2cSDA 21
// ESP32 ADC Pins
#define ADC1 32
#define ADC2 33
// Motor Sensor Pins
#define MOTSENS1 15
#define MOTSENS2 34
// Buttons
#define BTN1 14
#define BTN2 27
// Relay and Mosfet (BSP76) Pins
#define RELAIS1 18
#define RELAIS2 19
// LED1
#define LED1 5
```

```
// PWM1 and PWM2 Pins
#define PWM1 12
#define PWM2 13
// DAC Outputs 0..5V
#define DAC1 25
#define DAC2 26
// One Wire
#define DS18B20 4
```

In this code, a `ticker` object called `timer` is used to call a timer function every 10 milliseconds. A counter (`counter10ms`) is incremented by 1 with each call, and when this counter reaches the value 100, which corresponds to one second, it is reset. The variable `OneSecond` is set to `true` to indicate that one second has elapsed. The second counter (`CounterSeconds`) is used in loop() to call the output every second. OneSecond is set to false for the next run.

```
// Define 10ms Period
const unsigned long timerInterval = 10;
// Ticker-Objekt for the Timer-Funktion
Ticker timer;
// Counter
int counter10ms=0;
// When 1 second is elapsed
bool OneSecond = false;
// Counter for seconds
int CounterSeconds = 0;
// Timer-Funktion
// Called every 10 ms
void timerCallback()
{
 // Inc 10ms Counter
  counter10ms++;
  // If one Second
 if(counter10ms==100)
  {
     counter10ms = 0;
    // One Second
    OneSecond = true;
 }
}
```

This function sends data to a Nextion display to update the text of a textbox. It uses the serial interface `Serial2` to send a command that selects the textbox (`textbox`) and inserts the new text (`float`). The text is enclosed in quotes and the command is terminated with three terminating bytes (`0xFF`) to mark the end of the command. This enables the Nextion display to show the text correctly.

```
// Sends Data to Nextion Display in a textbox
void Nextion_Send_Text_to_Textbox(char *textbox,String sfloat)
{
        Serial2.write(textbox);
        Serial2.write(".txt=\"");
        Serial2.print(sfloat);
        Serial2.write("\"");
        Serial2.write(0xFF);
        Serial2.write(0xFF);
        Serial2.write(0xFF);
}
// Initializing the hardware
void setup()
{
  // Init USB UART
  Serial.begin(115200);
  // Nextion Display
  // TFTSerial.begin(9600);
  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);
  // Init Hardware
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(LED1, OUTPUT);
  pinMode(RELAIS1, OUTPUT);
  pinMode(RELAIS2, OUTPUT);
  pinMode(BTN1, INPUT);
  pinMode(BTN2, INPUT);
  // Clear Textboxes
  Nextion_Send_Text_to_Textbox("t1","Counter");
  Nextion_Send_Text_to_Textbox("t2"," ");
  Nextion_Send_Text_to_Textbox("t3"," ");
  Nextion_Send_Text_to_Textbox("t4"," ");
  Nextion_Send_Text_to_Textbox("t5"," ");
  Nextion_Send_Text_to_Textbox("t6"," ");
  Nextion_Send_Text_to_Textbox("txtT2"," ");
  Nextion_Send_Text_to_Textbox("txtT3"," ");
  Nextion_Send_Text_to_Textbox("txtT4"," ");
  Nextion_Send_Text_to_Textbox("txtT5"," ");
```

```
Nextion_Send_Text_to_Textbox("txtT6"," ");
// Init Timer with 10ms
timer.attach_ms(timerInterval, timerCallback);
}
```

The `loop()` function checks whether a second has elapsed based on the `OneSecond` variable. If this is the case, `OneSecond` is reset to `false` and the `CounterSeconds` counter is incremented by one. The new value of `CounterSeconds` is sent to the serial monitor via the serial interface and sent to a text box on the Nextion display. In addition, the status of two LEDs is switched: the internal LED and another LED (`LED1`). The code thus ensures regular updates and visual feedback via the LEDs.

```
void loop()
{
  // Check if one second has passed
  if (OneSecond) {
    // Reset the boolean variable OneSecond
    OneSecond = false;
    CounterSeconds++;
    // Sends variable to USB UART and Seriell Monitor
    Serial.print("Timer: ");
    Serial.println(CounterSeconds);
    // Sends variable to display
    Nextion_Send_Text_to_Textbox("txtT1",String(CounterSeconds));
    // Switching LED
    digitalWrite(LED_BUILTIN,!digitalRead(LED_BUILTIN));
    digitalWrite(LED1,!digitalRead(LED1));
 }
  // Additional code for the main loop
}
```

#### 1.10 Program example with the ESP32

In the next program, the buttons are queried. The button on BTN1 is queried and debounced with the function read\_btn1(). In the timer function, which is called every 10 ms, the pushbutton at BTN2 is read. If BTN2 is at low level three times in succession, i.e., 30 ms, the key\_press variable is set to True. The key press is queried with the get\_key() function. When the information is pressed, the values of the state variables state1 or state2 are changed to save the current state and then switch the RELAIS1 and RELAIS2 outputs. Press once to switch on, press again to switch off. These functions will be used later to set the setpoint of controllers.

#### Listing 2: Arduino\_TimerTick\_USB\_Hardware.ino

```
//--
11
11
            Arduino Timer Test
             _____
11
11
       Timer Testprogramm for Arduino ESP32 PID Fuzzy Control Board
11
       10ms Timer Interrupt Function
11
        Read 2 Buttons and switch Relais Pin with BTN1 and LED1 with BTN2
11
11
// Date: 08/05/2024
// Author: Josef Bernhardt
// Hardware: Arduino (ESPRESSIF) ESP32
11
// File: Arduino_TimerTick_USB_Hardware.ino
//-----
#include <Arduino.h>
#include <Ticker.h>
int LED_BUILTIN = 2;
// ESP32 UART 2 Pins
#define RXD2 16
#define TXD2 17
// I2C Pins
#define I2CSCL 22
#define I2cSDA 21
// ESP32 ADC Pins
#define ADC1 32
#define ADC2 33
// Motor Sensor Pins
#define MOTSENS1 15
#define MOTSENS2 34
// Buttons
#define BTN1 14
#define BTN2 27
// Relay and Mosfet (BSP76) Pins
#define RELAIS1 18
#define RELAIS2 19
// LED1
#define LED1 5
```

```
// PWM1 and PWM2 Pins
#define PWM1 12
#define PWM2 13
// DAC Outputs 0..5V
#define DAC1 25
#define DAC2 26
// One Wire
#define DS18B20 4
// Define 10ms Period
const unsigned long timerInterval = 10;
// Ticker-Objekt for the Timer-Funktion
Ticker timer;
// Counter 10 for Timer
int counter10ms=0;
// When 1 second is elapsed
bool OneSecond = false;
// State for Buttons
int state1 = 0;
int state2 = 0;
// Variable for BTN2 in Timer
int key_state = 0;
int key_counter = 0;
int key_press = 0;
```

In addition to the time measurement, the status of the BTN2 button is queried in the timer callback function. To ensure more stable button detection, the button must be pressed for a total of at least 30 milliseconds. When this counter is reached, the push-button status is updated, and key\_press is set to 1 if the push-button is pressed for 30 ms. This method ensures that the button is actually pressed for a certain time before it is recognized as pressed.

```
// Timer-Funktion
// Called every 10 ms
void timerCallback()
{
    // Inc 10ms Counter
    counter10ms++;
    // If one Second
    if(counter10ms==100)
    {
```