Programmhandbuch Akkusimulator

Inhaltsverzeichnis

[Datei tools\_V21.py 1](#_Toc108085396)

[Klasse SensorRead(ttk.Frame) 1](#_Toc108085397)

[Klassenvariablen 1](#_Toc108085398)

[Klassenmethoden 1](#_Toc108085399)

[Klasse Countdown(ttk.Frame) 3](#_Toc108085400)

[Klassenvariablen 3](#_Toc108085401)

[Klassenmethoden 3](#_Toc108085402)

[Klasse EepromControl() 3](#_Toc108085403)

[Klassenvariablen 3](#_Toc108085404)

[Klassenmethoden 3](#_Toc108085405)

[Klasse HexValConvert() 5](#_Toc108085406)

[Datei EepromData.py 6](#_Toc108085407)

[Listen 6](#_Toc108085408)

# Datei tools\_V21.py

## Klasse SensorRead(ttk.Frame)

Zuständig für die Kommunikation mit dem Strom/Spannungsmesschip INA226. Die Klasse wird beim Aufruf mit einer Anzeige initialisiert und kann direkt in die Benutzerobefläche integriert werden.

### Klassenvariablen

|  |  |
| --- | --- |
| **Klassenvariable** | **Funktion** |
| Ina226\_adress | I2C Adresse des INA226 |
| Ina226 | Initialisiert I2C Kommunikation |
| currentLSB | Benötigt für Berechnung des Stroms |
| Cal | Wert Kalibrierungsregister |
| maxExpCurr | Maximal erwarteter Strom |
| shuntResValue | Wert des verwendeten Widerstands in Ohm |
| busVoltOffset  shuntVoltOffset  busCurrOffset  powerOffset | Kalibrierwerte für Messwerte |

### Klassenmethoden

|  |
| --- |
| ina226\_readReg: wird genutzt, um spezifisches Register des INA226 auszulesen |
| **Parameter**  **adress**: auszulesendes Register des INA226  **mögliche Werte:**  self.ina226\_config\_reg = 0 #R/W  self.ina226\_shunt\_reg = 1 #R  self.ina226\_bus\_reg = 2 #R  self.ina226\_power\_reg = 3 #R  self.ina226\_curr\_reg = 4 #R  self.ina226\_cal\_reg = 5 #R/W |
| **Rückgabewert**  16 Bit uint Registerwert |

|  |
| --- |
| ina226\_writeReg: wird genutzt, um spezifisches Register des INA226 zu beschreiben |
| **Parameter**  **adress**: zu beschreibendes Register des INA226  **mögliche Werte:**  self.ina226\_config\_reg = 0 #R/W  self.ina226\_cal\_reg = 5 #R/W    **content:** neuer Wert des Registers  **mögliche Werte:**  0…65535 🡪 16 Bit |
| **Rückgabewert**  - |

|  |
| --- |
| ina226\_calibrateReg: wird genutzt, um Kalibrierregister zu beschreiben  *für mehr Infos siehe INA226 Datenblatt* |
| **Parameter**  **maxExpectCurr**: maximal erwarteter Strom  **mögliche Werte:**  0…20  **rShunt**: Wert des verwendeten Shunts  **mögliche Werte:**  0.001-1(Wert des Widerstands in Ohm) |
| **Rückgabewert**  - |

**Get/Set-Funktionen sind in der Klasse für alle Klassenvariablen vorhanden.**

## Klasse Countdown(ttk.Frame)

Erzeugt ein Widget mit Timer, welcher einen Countdown mit vorgegebener Zeit startet.

### Klassenvariablen

|  |  |
| --- | --- |
| **Klassenvariable** | **Funktion** |
| Ina226\_adress | I2C Adresse des INA226 |
| Dur | Timervariable |
| durStart | Startzeit |
| secFormat | Formatierter Counter(zeigt immer 2 Stellen an) |

### Klassenmethoden

|  |
| --- |
| Countdown:  Wird genutzt, um die Startzeit beim Countdown zu setzen und den Timer an sich zu starten |
| **Parameter:**  **-** |
| **Rückgabewert:**  **-** |

|  |
| --- |
| startCountdown:  in dieser Funktion wird der timer dekrementiert und das Label **tl** aktualisiert |
| **Parameter:**  **-** |
| **Rückgabewert:**  **-** |

**Get-Methoden sind für die Variablen dur und durStart vorhanden.**

# Klasse EepromControl()

Die Klasse EepromControl steuert die Kommunikation über UART, vor allem mit dem Arduino.

### Klassenvariablen

|  |  |
| --- | --- |
| **Klassenvariable** | **Funktion** |
| sendBuffer | Buffer für ausgehende Daten |
| receiveBuffer | Buffer für eintreffende Daten |
| Ser | Initialisiert UART-Kommunikation |

### Klassenmethoden

|  |
| --- |
| sendPackage:  versenden eines einzelnen 5\*8-Bit UART-Pakets |
| **Parameter:**  Id:  Mögliche Werte:  Siehe EepromData.py --> uartCMD  Adress:  Mögliche Werte:  0…255  Content:  Mögliche Werte:  0…255 |
| **Rückgabewert:**  **-** |

|  |
| --- |
| receivePackage:  empfangen eines einzelnen 3\*8-Bit UART Pakets |
| **Parameter:**  **-** |
| **Rückgabewert:**  Uint8 payload |

|  |
| --- |
| readSingleRegister:  auslesen eines Registers vom Arduino |
| **Parameter:**  **Adress**  Mögliche Werte:  0…255 |
| **Rückgabewert:**  Uint8 Registerwert |

|  |
| --- |
| readAllRegisters:  aktualisieren aller Registerwerte auf dem Raspberry Pi, Werte werden vom Arduino übernommen |
| **Parameter:**  **-** |
| **Rückgabewert:**  - |

|  |
| --- |
| writeSingleRegister:  beschreiben eines Registers vom Arduino |
| **Parameter:**  **Adress**  **Mögliche Werte:**  0…255  **Content:**  **Mögliche Werte:**  0…255 |
| **Rückgabewert:**  - |

|  |
| --- |
| readNTC:  auslesen des NTC-Wertes auf dem Arduino |
| **Parameter:**  - |
| **Rückgabewert:**  Uint8 NTC-Wert |

|  |
| --- |
| writeNTC:  setzen des NTC-Wertes auf dem Arduino |
| **Parameter:**  - |
| **Rückgabewert:**  **-** |

|  |
| --- |
| readOverVoltage:  auslesen des Überspannungsbytes vom Arduino |
| **Parameter:**  - |
| **Rückgabewert:**  Uint8 Überspannungswert |

|  |
| --- |
| wrtieOverVoltage:  setzen des Überspannungsbytes auf dem Arduino |
| **Parameter:**  **-** |
| **Rückgabewert:**  - |

|  |
| --- |
| setEeprom:  initialisieren aller Eepromwerte auf dem Raspberry Pi |
| **Parameter:**  **-** |
| **Rückgabewert:**  - |

## Klasse HexValConvert()

Ist noch nicht fertig geschrieben, soll genutzt werden, um die Hex-Werte aus dem Eeprom in physikalische Werte zu „übersetzen“. Die Klasse ist noch nicht fertig geschrieben und muss noch ergänzt werden.

# Datei EepromData.py

Zentrale Datei zum speichern von Eeprom Daten und Listen der einzelnen Befehle für UART- und I2C-Operationen

## Listen

|  |  |
| --- | --- |
| Listenname | Funktion |
| uartCMD | Command Bytes für UART-Kommunikation |
| EepromDataDict | Eeprom-Werte, die mehrmals vorkommen  Kann genutzt werden, um Werte zu initialisieren/resetten |
| InfoDataDict | Kann genutzt werden, um Werte zu initialisieren/resetten |
| EepromDataComplete | Beinhaltet Eeprom-Daten |
| InfoData | Beinhaltet Daten für Überspannung/NTC-Wert |
| EepromDataSafety | Derzeit ungenutzt, kann für spezifische Darstellungen des Eeprom genutzt werden |
| EepromDataCharge | kann für spezifische Darstellungen des Eeprom genutzt werden |
| EepromDataCycle | kann für spezifische Darstellungen des Eeprom genutzt werden |
| EepromDataValues | Wird genutzt, um Werte des Eeproms zu ändern  Array mit Werten 0-255(8 Bit) |
| CalibrationFine | Wird genutzt, um Werte des INA226 zu kalibrieren  Array mit Werten 0-1 mit 0.1 Schrittweite |
| CalibrationCoarse | Wird genutzt, um Werte des INA226 zu kalibrieren  Array mit Werten 0-6 |
| NTCTempValues | Dezimalwerte des NTC |
| NTCTemps | Temperaturen des NTC |

# Arduino-Programm

## Datei EepromData.cpp

Diese Datei beinhaltet alle Funktionen zum Setzen verschiedener Zustände des Arduino-Eeproms.

Die Funktionen wurden zuerst in **EepromData.h**  initialisiert.

### Funktionen

|  |
| --- |
| **setEeprom:**  wird genutzt, um die Werte des Eeprom auf dem Arduino zu initialisieren  ohne diese Funktion stehen keine sinnvollen Daten im Arduino-Eeprom |
| **Parameter:**  - |
| **Rückgabewerte:**  - |

|  |
| --- |
| **setInfoData:**  wird genutzt, um die Werte von Überspannung und NTC zu initialisieren  ohne diese Funktion stehen keine sinnvollen Daten in den jeweiligen Buffern |
| **Parameter:**  Ntc:  Mögliche Werte: 0…255 🡪 siehe NTC-Wertetabelle  Ov:  Mögliche Werte: 0…255 🡪 siehe OV-Wertetabelle |
| **Rückgabewerte:**  - |

|  |
| --- |
| **activeChargeParameter:**  wird genutzt, um die aktiven Ladeparameter zu aktivieren/deaktivieren |
| **Parameter:**  Bool yes:  Mogliche Werte: true 🡪 aktive Ladeparameter werden gesetzt  False🡪aktive Ladeparameter werden deaktiviert |
| **Rückgabewerte:**  - |

|  |
| --- |
| **setActiveChargeParameter:**  wird genutzt, um die Werte der aktiven Ladeparameter im Eeprom auf dem Arduino zu initialisieren  ohne diese Funktion stehen keine sinnvollen Daten im Arduino-Eeprom |
| **Parameter:**  - |
| **Rückgabewerte:**  - |

|  |
| --- |
| **setEepromToSmallState:**  das Eeprom wird als Eeprom in kleiner Ausführung(128 Bytes) initialisiert |
| **Parameter:**  - |
| **Rückgabewerte:**  - |

|  |
| --- |
| **setEepromToNormalState:**  Das Eeprom wird in normaler Ausführung(256 Bytes) initialisiert  **Standardversion, Defaultstatus des Eeproms** |
| **Parameter:**  - |
| **Rückgabewerte:**  - |

## Datei main.cpp

Hauptdatei des auf dem Arduino laufenden Programms. Beinhaltet alle Funktionen zu Initialisierung aller Eeprom-Werte, Antwort auf einkommende Befehle und Kommunikation mit dem Ladegerät und dem Raspberry Pi.

### Funktionen

|  |
| --- |
| receiveEvent**:**  Event, welches ausgelöst wird, wenn das Ladegerät(I2C-Master) Daten an den Arduino sendet.  Abhängig von der übergebenen Adresse werden unterschiedliche Funktionen ausgeführt  **EINE VERÄNDERUNG DIESES CODES KANN DAZU FÜHREN, DASS DAS LADEGERÄT NICHT MEHR MIT DEM ARDUINO KOMMUNIZIEREN KANN!** |
| **Parameter:**  - |
| **Rückgabewerte:**  - |

|  |
| --- |
| requestEvent**:**  Event, welches ausgelöst wird, wenn das Ladegerät(I2C-Master) Daten vom Arduino anfordert.  Abhängig von der übergebenen Adresse werden unterschiedliche Funktionen ausgeführt  **EINE VERÄNDERUNG DIESES CODES KANN DAZU FÜHREN, DASS DAS LADEGERÄT NICHT MEHR MIT DEM ARDUINO KOMMUNIZIEREN KANN!** |
| **Parameter:**  - |
| **Rückgabewerte:**  - |

|  |
| --- |
| Uart\_sendPackage**:**  Funktion zum senden von Paketen über UART  **Derzeit genutzt, um Pakete an Raspberry Pi zu senden** |
| **Parameter:**  Uint8\_t payload  Mögliche Werte: 0…255 |
| **Rückgabewerte:**  - |

|  |
| --- |
| Uart\_readSingleRegister**:**  Sendet den Wert eines einzelnen Registers zurück an den Raspberry Pi |
| **Parameter:**  Uint8\_t payload  Mögliche Werte: 0…255 |
| **Rückgabewerte:**  - |

|  |
| --- |
| Uart\_writeSingleRegister**:**  setzt den Wert eines einzelnen Registers im Arduino-Eeprom |
| **Parameter:**  Uint8\_t adress:  Mögliche Werte: 0…255  Uint8\_t content:  Mögliche Werte: 0…255 |
| **Rückgabewerte:**  - |