paaalm07

[Email address]

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Motor-Prüfstand STEUERUNG

[Document subtitle]

Contents

[Einführung 2](#_Toc172334874)

[Konzept 2](#_Toc172334875)

[Hardware 2](#_Toc172334876)

[Elektronik Komponenten 2](#_Toc172334877)

[Zusammenbau 2](#_Toc172334878)

[Software 2](#_Toc172334879)

[Implementierung 3](#_Toc172334880)

[Tools 3](#_Toc172334881)

[Nutzung 3](#_Toc172334882)

[Verbesserungspotential 3](#_Toc172334883)

# Einführung

Erfassung von Sensoren bei einem Motorprüfstand und Anzeige auf einem PC.

Die Sensoren sollen verschiedene Spannungen, Drehzahlen und Temperaturen erfassen.

Zündzeitpunkt und Kurbelwellenwinkel sind auch zu erfassen.

Luftmassenmesser und Klopfsensor wären auch wünschenswert.

# Konzept

RaspberryPi als zentraler „Messsystem Server“, sowohl für Messungen als auch für Steuerungen.  
Es gibt verschiedene Adapter Platinen (HATs) für den RaspberryPi aus der Digilent MCC Serie.

* MCC118/128 – Spannungsmessung
* MCC134 – Temperaturmessung via Thermocouple (zB K-DUT)
* MCC152 – Spannungsausgang und DIO
* MCC172 – IEPE (Integrated Electronics Piezo Electric) Messungen.

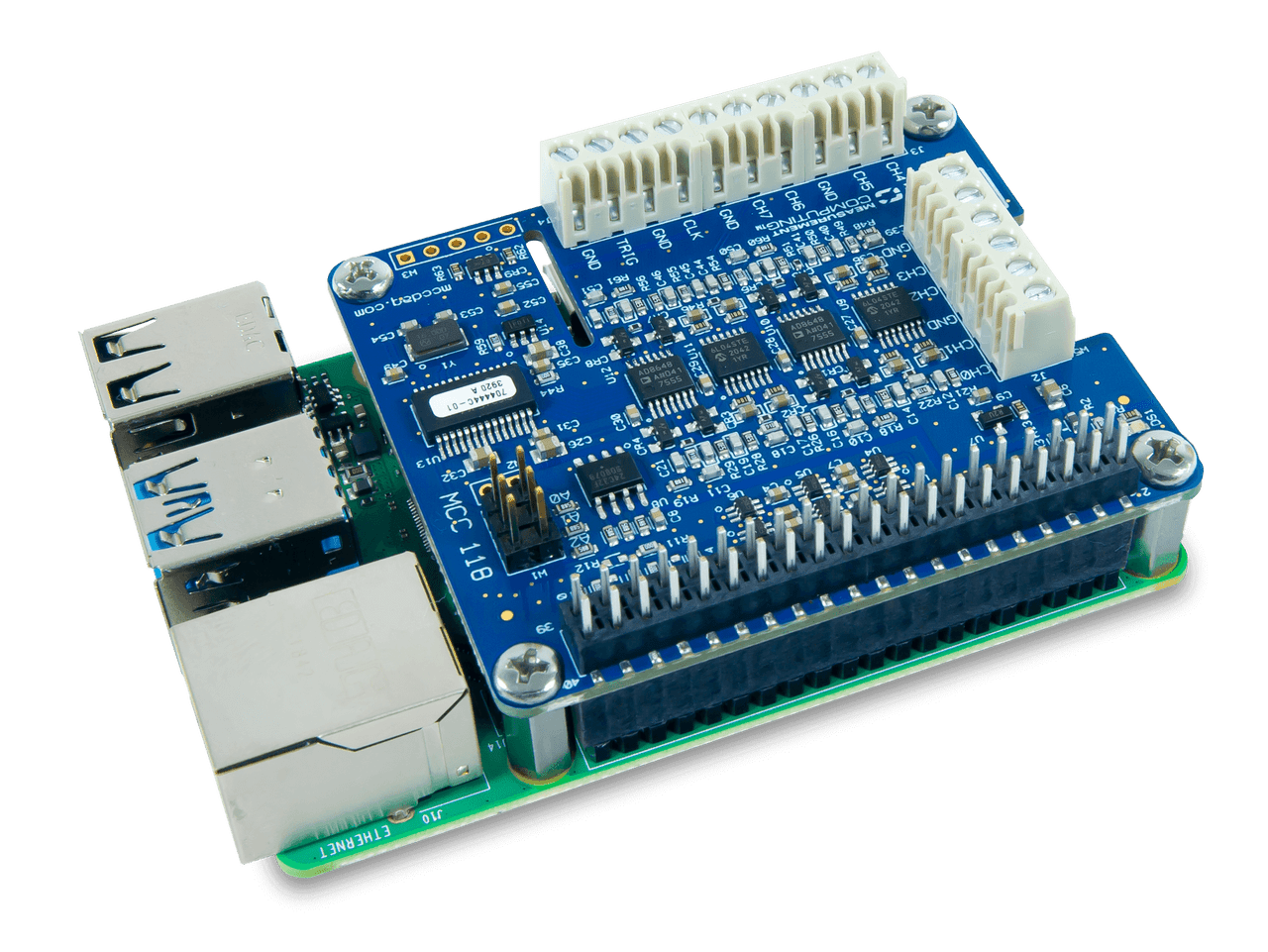
Bis zu 8 HATs können gestapelt werden für einfache Erweiterung.

Zur Visualisierung des Messsystems wird die „ComVisu“ Software verwendet. Sowohl als Anzeige der Messdaten als auch zur Steuerung der Aktoren.

Die Messdaten werden nur per Software getriggert – kein Echtzeit System. Ausreichend weil die ComVisu nur begrenzte Anzahl an Kommandos verarbeiten kann, ansonsten Packet Überlauf.

## Hardware

### Elektronik Komponenten

* RaspberryPi 5 8GB mit aktiver Kühlung
* Digilent MCC118 HAT zur Spannungsmessung  
  
* Anschluss Board für Digitale IOs inkl. Statusanzeige
* Schrittmotor & Treiber („Bremspedal“)
* Servo Motor („Gas“)

### Zusammenbau

Alles zusammen wird in ein Gehäuse „fliegend“ verkabelt. Sollte es zu viele Störungen geben wird sich nachträglich darum gekümmert. Zum Anschluss der Sensoren werden im Gehäuse verbaut.

## Software

### Python

SourceCode wird in Python 3.11 entwickelt.

* Server für Messungen (RPI 🡪 ComVisu)
* Client für Commando Verarbeitung (ComVisu 🡪 RPI)
* Driver Implementierung
  + RaspberryPi Klasse
    - Frequenzzähler Klasse
    - Input Klasse
    - Output Klasse
  + Digilent MCC118 Klasse
    - VoltageMesurement Klasse
  + SchrittMotor Klasse
  + ServoMotor Klasse
* Möglichkeit zur Klaibrierung von Sensoren

### ComVisu

* „Keep Alive Check“ des Mess-Systems
* Anzeige von Messdaten in Diagrammen
* Start/Stop der Messung
* Schrittmotor Ansteuerung („Bremspedal“)
* Servomotor Ansteuerung („Gas“)

# Implementierung

## Tools

Win32Diskimager, Visual Studio Code, ComVisu

SVN: Version Control System für Windows

GIT: Version Control System für RaspberryPi

# Nutzung

# Verbesserungspotential