Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

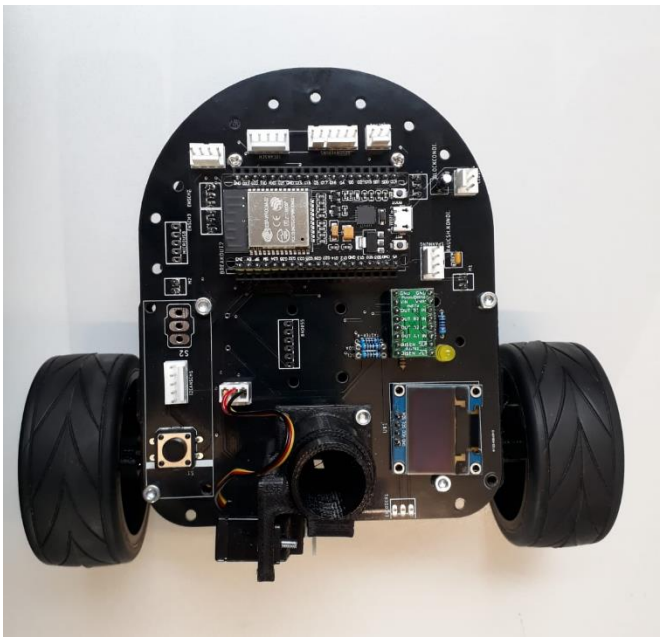
Näher dran.

MMT Fakultät für Maschinenbau und Mechatronik

eMalRob-FG

Projekt: eMalRob Austausch

Detaillierte Anleitung der Konfiguration und erstmalige Inbetriebnahme anhand verschiedener Szenarien



Projektbearbeiter:		
Fabian Passler	pafa1011	56680
Florian Herker	hefl1018	53667

Modulbezeichnung: ***MECB710 Informationstechnik***

Sommersemester 2020

– abgegeben am 28. Mai 2020 –

Betreuender Professor:
Prof. Dipl.-Ing. Jürgen Walter
Hochschule Karlsruhe – Technik und Wirtschaft
Fakultät für Maschinenbau und Mechatronik
Moltkestr. 30; 76133 Karlsruhe

Inhaltsverzeichnis

Vorwort

Guten Tag liebe Leser und Programmierneugierige allen Alters,

in diesem Dokument finden Sie eine sowohl strikte als auch detaillierte Anleitung den sogenannten eMalRob-FG auf verschiedene Weisen zu konfigurieren und in Betrieb zu nehmen.

Bei dem eMalRob-FG handelt es sich um einen von Studenten entwickelten Mal Roboter, welcher Menschen jedes Alters die Basis das Programmieren auf spielerische Weise beibringen und aufzeigen soll. Zielgerichtet auf Schüler wurde der eMalRob-FG nach der Vorlage des sogenannten Malroboter des Unternehmens Fischertechnik konstruiert und entwickelt. Ein wesentlicher Unterschied hierbei ist jedoch, dass ein großes Augenmerk auf den Aspekt Anschaffungskosten gelegt wurde. Der Ihnen im Dokument erläuterte Roboter ist um einen signifikanten Wert kostengünstiger als der als Vorlage dienende Roboter von Fischertechnik.

Anbei können Sie durch vier verschiedene Szenarien wählen wie Sie Ihren Roboter programmieren möchten und werden anhand eines von uns ausarbeiteten „roten Fadens“ von A-Z mit Hilfe von Bildern und Anleitungstexten bis an ihr Ziel geführt.

Ebenso wurde darauf geachtet, dass sämtliche Infos, zusätzliche Programme, Datenstände als auch Anleitungsverweise etc. sich auf einem bestimmten Server befinden. Somit ist die leichte als auch umstandsfreie Beschaffung aller wichtigen und interessanten Inforationen für Sie sichergestellt.

Viel Spaß beim Programmieren und beim spielerischen Weiterbilden.

Beschreibung einzelner Konfiguration- & Inbetriebnahme-Szenarien

Szenario 1 – Cody++ Programmierung direkt auf dem eMalRob-FG

In diesem Szenario handelt es sich wie bereits im Titel angemerkt um eine Programmierung des eMalRob-FG über den auf dem Roboter montierten und installierten ESP32. Diese Programmierung findet direkt auf dem ESP statt. Anhand der Programmieroberfläche, in der Sie letztendlich Ihr Programm mit kleinen Programmbausteinen spielerisch zusammensetzten können, ist der ESP in der Lage über eine von sich geöffneten WIFI Verbindung das vollendete Programm abzuspielen und somit umzusetzen. Folge dessen wird Ihr Roboter genau dies Programm abarbeiten, welches Sie auf Ihrem Rechner über die Programmieroberfläche Cody++ ausgearbeitet und über das vom ESP32 geöffneten WIFI übertragen haben.

Szenario 2 – Cody++ Programmierung extern auf dem Rechner

Hierbei handelt es sich um eine Programmierung die Sie ebenso auf der Programmieroberfläche Cody++, welche sie auf dem HIT-Server der HS Karlsruhe finden, durchführen. Jedoch wird nun das fertiggestellte Programm entweder per Serielle Schnittstelle (USB) oder per WIFI (Router) übertragen und somit auf den eMalRob-FG bzw. den ESP32 geschrieben.

Szenario 3 – Fernsteuerung des eMalRob via Smartphone-App

Mithilfe einer von Studenten eigens entwickelten App ist es ebenso möglich sich mit dem eMalRob-FG sowohl zu konfigurieren als auch diesen zu steuern. Hierbei behilft sich die App über eine „direkte“ Steuerung anhand der in Ihrem Smartphone vorhandenen Sensoren. Dies bedeutet, dass der Roboter in der Lage ist verschiedene Bewegungsmuster zu fahren, wenn Sie zum Beispiel das Smartphone in eine gewisse Richtung neigt. Sie können sich Ihr Smartphone wie eine Fernsteuerung im Kleinformat vorstellen und diese derartig nutzen.

Szenario 4 – Freie Programmierung extern auf dem Rechner

Das letzte und auch aufwendigste Szenario ist die EVA ESP Programmierung, da diese ohne jegliche Hilfsunterstützung an Programmteilcodes fungiert. Das bedeutet, dass Sie eine bei der bereits erwähnten Cody++ ähnlichen Programmieroberfläche benutzen, um ebenso durch Bausteinzusammensetzung ein Programm zu erstellen. Jedoch besitzt die EVA ESP Programmieroberfläche keine Hilfsmittel, was bedeutet, dass Sie den gesamten C++ Code mit allen dazugehörigen Befehlen, wie zum Beispiel Bibliotheken, weiteren Cpp & Header-Dateien selbst schreiben müssen. Diese Variante wird definitiv nicht als Einsteigervariante empfohlen.

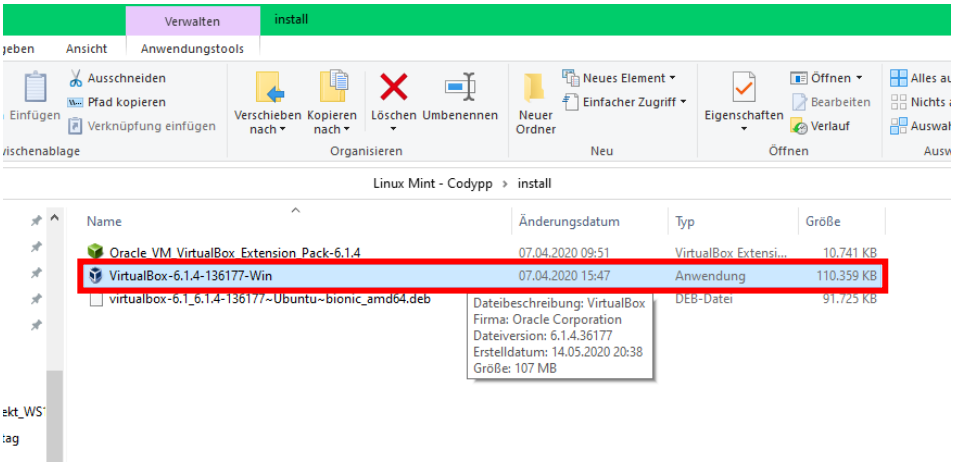
Detaillierte Anleitung der Konfiguration und Inbetriebnahme anhand der Szenarien

S1 – Cody++ Programmierung direkt auf dem eMalRob-FG

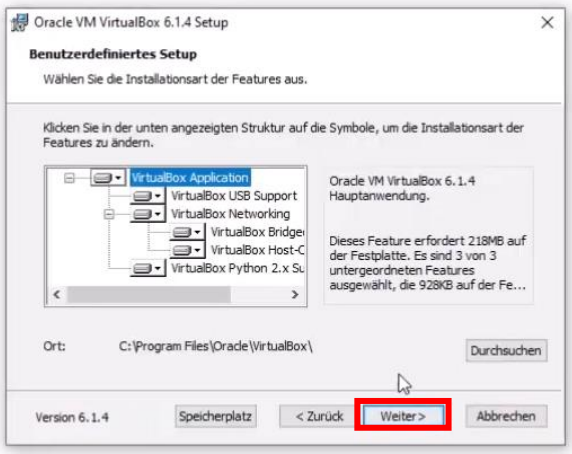
Der erste Schritt ist das Downloaden des Zip Ordners auf GitHub. Dieser Ordner dient als komplette Zusammenfassung aller benötigten Dateien.

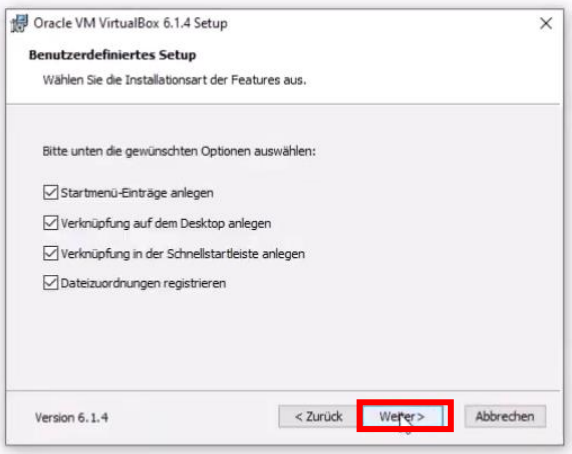
Zip Dateien von GitHub sind unter folgendem Link erhältlich:
https://github.com/Mechatronikwelt/eMalRob-FG.git

Der erste Schritt ist die Installation der Oracle VirtualBox. Auf der GitHub-Seite finden Sie die benötigten Informationen, wo sich die zu verwendenden Dateien („Linux Mint - Codypp.zip“) befinden. Downloaden, entpacken und führen Sie die Datei aus, welche Sie im Unterordner *„/install“* finden. Die *.exe* Datei benutzen Sie für ein Windows Betriebssystem, bzw. die *.deb* Datei für ein Linux Betriebssystem.

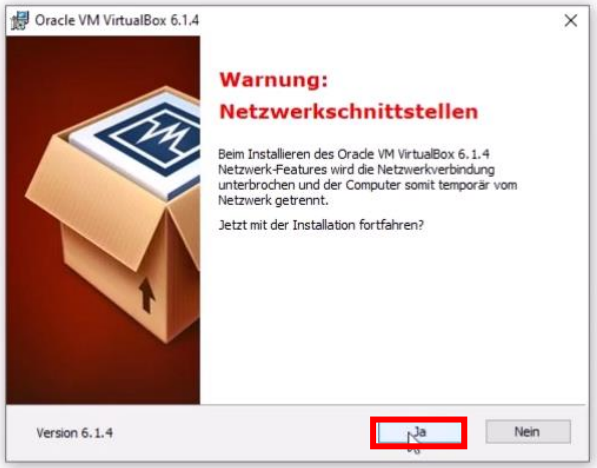


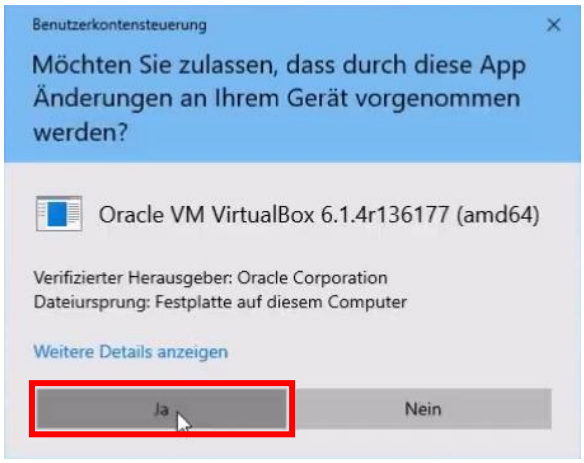
Das folgende „Benutzerdefinierte Setup“ jeweils mit *„weiter“* bestätigen.

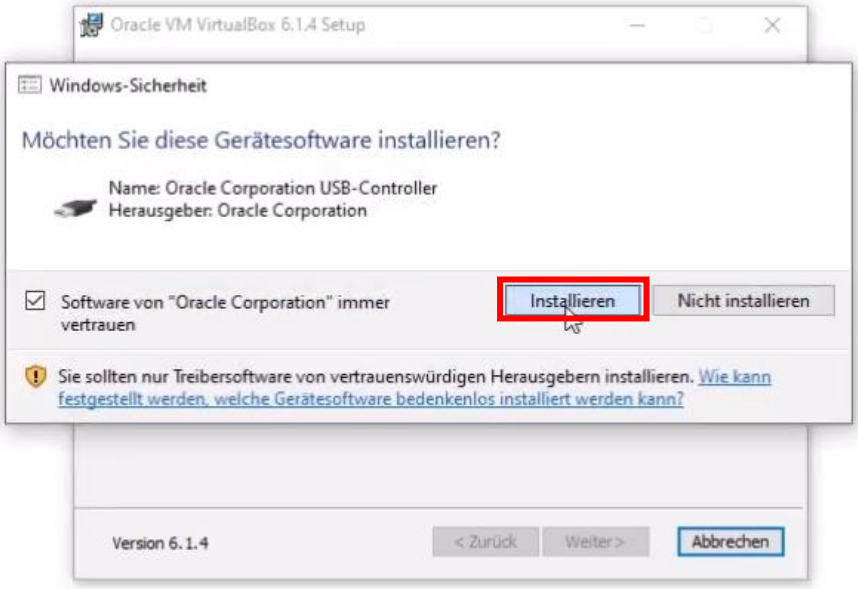




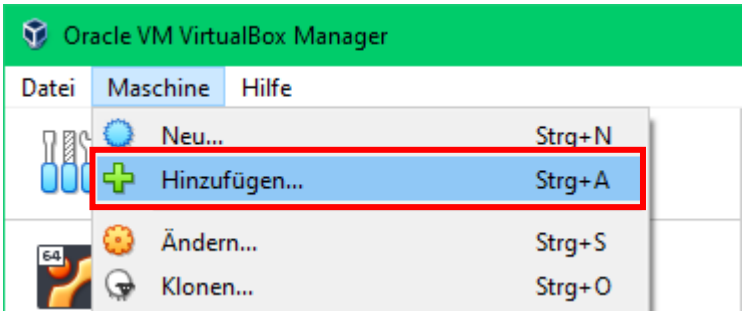
Die Netzwerkschnittstellenwarnung mit *„Ja“* bestätigen, um die Installation fortzufahren. Bestätigen Sie die Installation Oracle VM VirtualBox als Admin mit *„Ja“* und Gerätesoftware mit *„installieren“*.



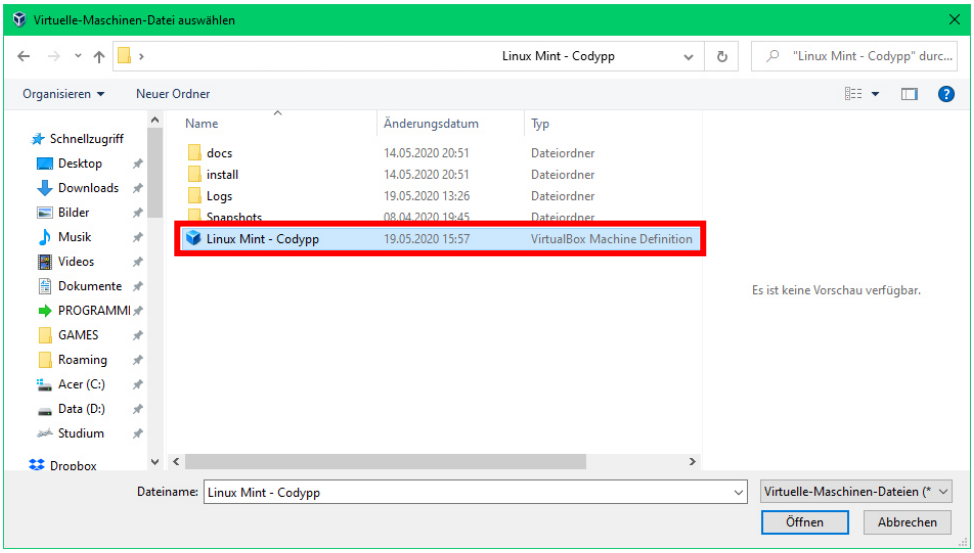


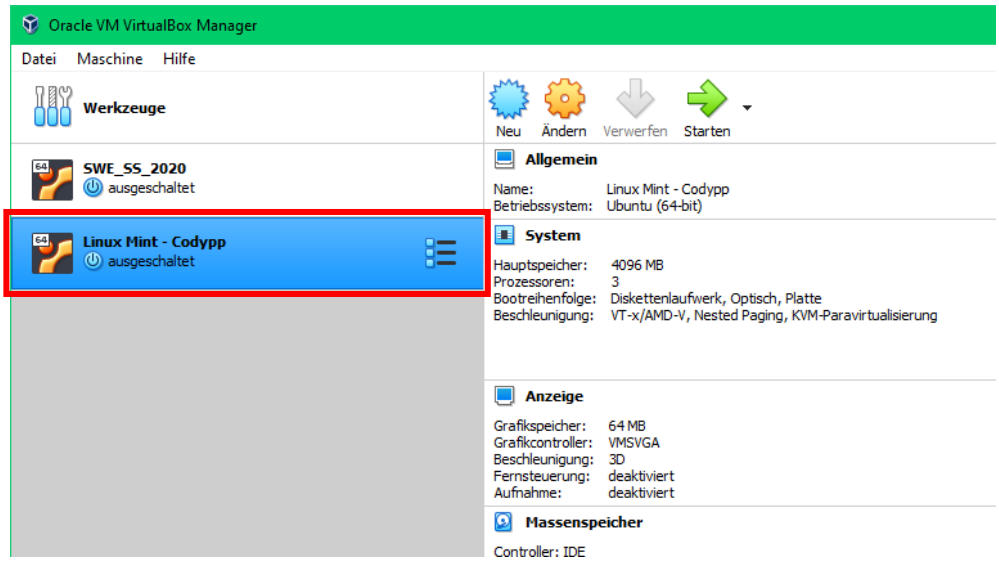


Nun ist die Installation der Oracle VM Virtual Box vollendet und abgeschlossen.
Fügen Sie jetzt die benötigte Virtuelle Maschine unter *„Maschine -> Hinzufügen“* hinzu.



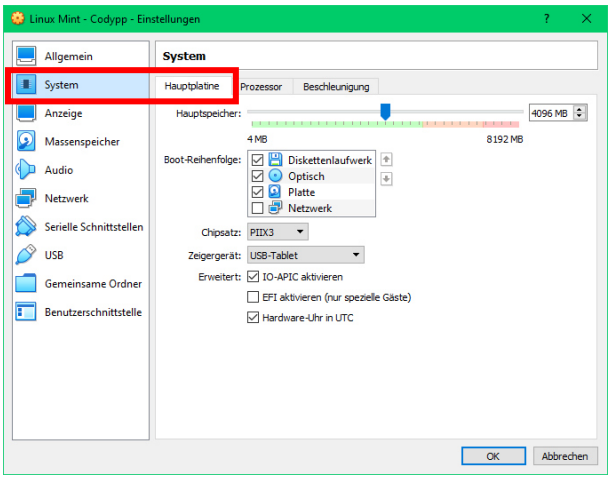
Anschließend wird das Betriebssystem im *„Linux Mint – Codypp\“* Ordner gesucht und die „Linux Mint – Codypp.vbox“ ausgeführt. Somit wird die benötigte VM hinzugefügt und ist zum Starten bereit.

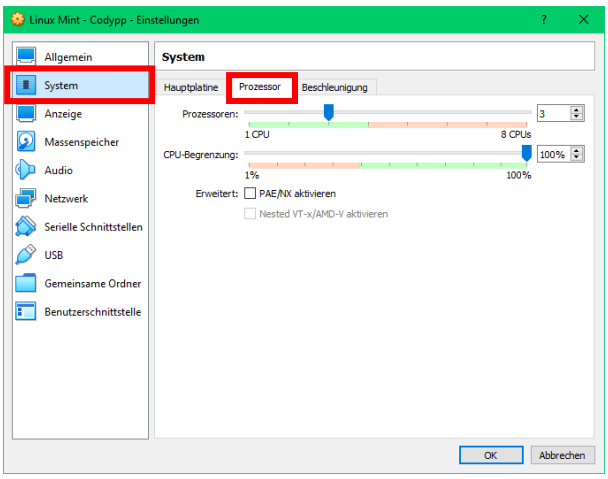


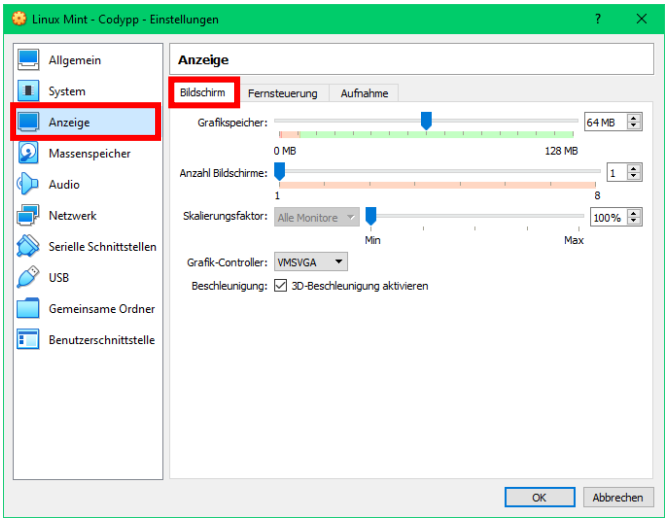


Passen Sie nun die Einstellungen der Virtuellen Maschine unter *„Maschine -> Ändern“* an.

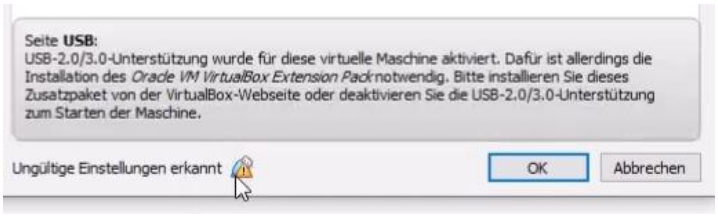
Hierbei müssen Sie je nach PC folgende Anpassungen ähnlich den nachfolgenden Bildern tätigen.



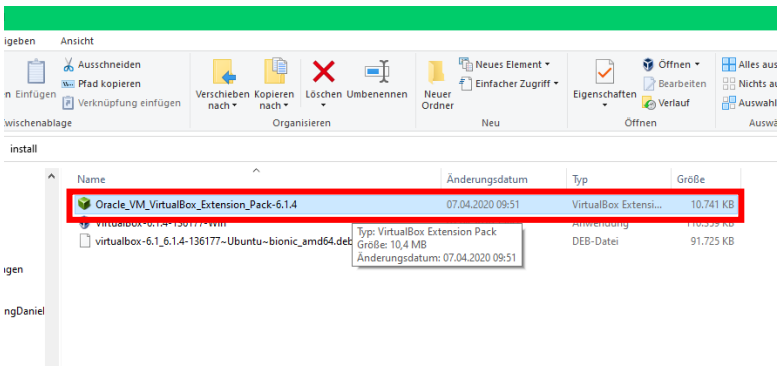




Hinweis: Sollte folgende Fehlermeldung im unteren Bereich des Fensters auftauchen, muss das „Extension Pack“ installiert werden.

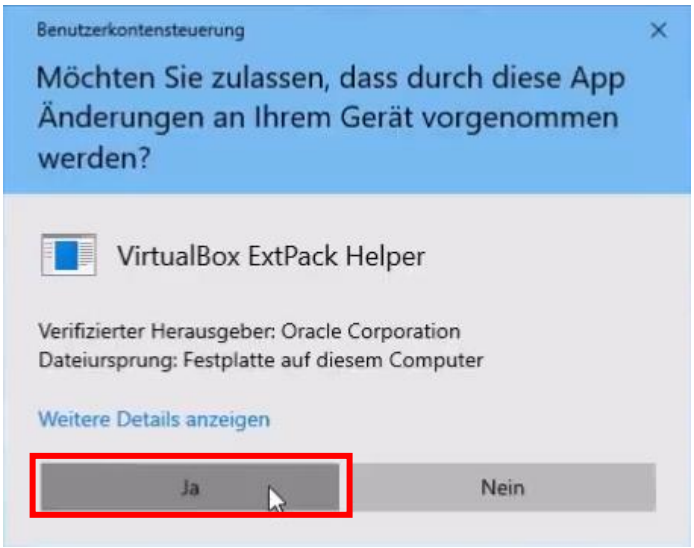


Die Datei zur Installation des „Extension Pack“ befindet sich in dem bereits zuvor verwendeten „install“ Ordner.

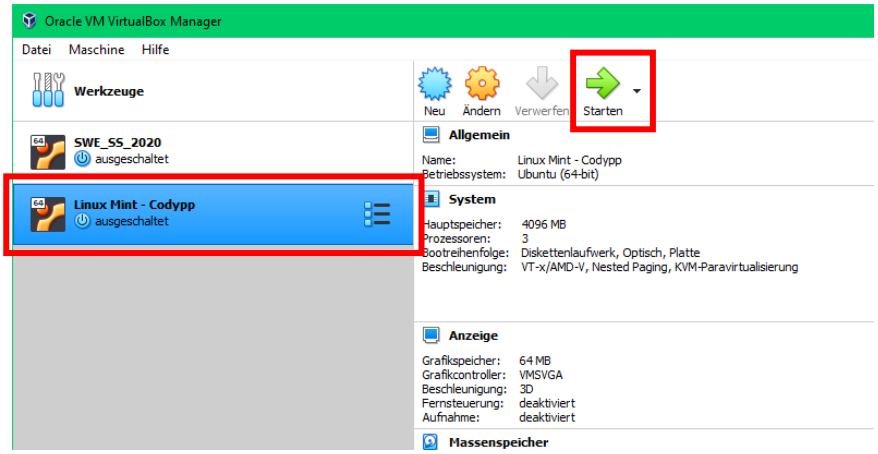


Durch Doppelklick wird die Installation begonnen. Durch *„Installieren“* wird die Installation bestätigt.

Bestätigen Sie die Installation des „Extension Pack“ als Admin mit *„Ja“* und beenden Sie die Installation mit *„OK“*.

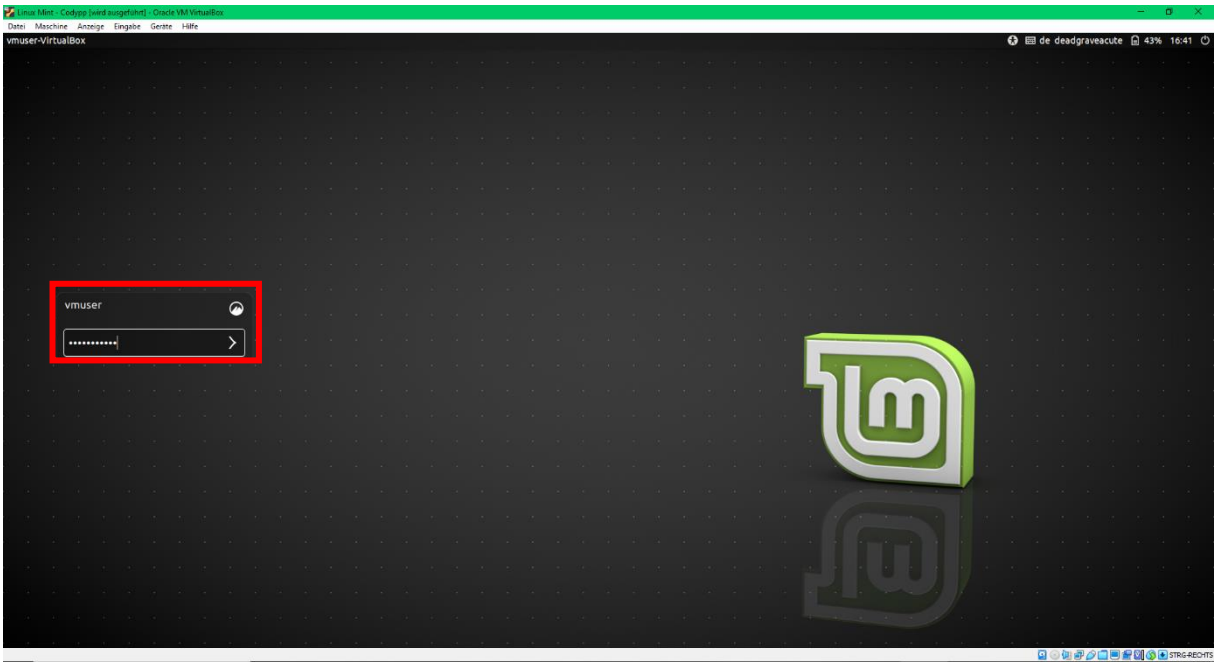


Starten Sie bitte nun die Virtuelle Maschine anhand des grünen „Start“ Pfeils.

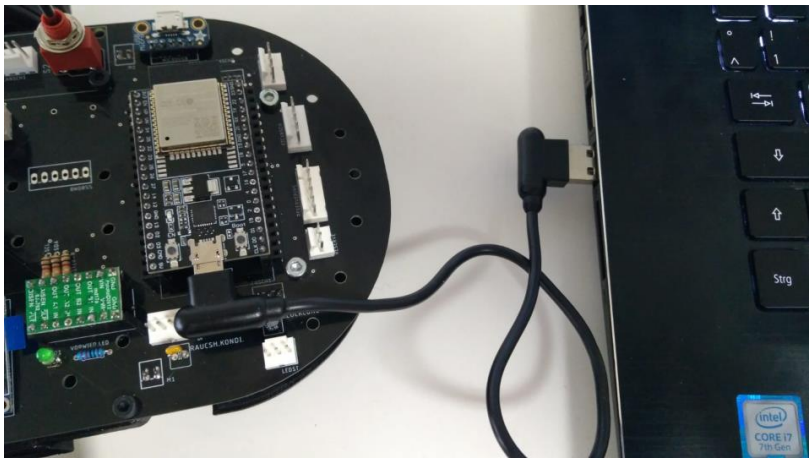


Nach Start der Virtuellen Maschine loggen Sie sich bitte mit folgenden Daten ein:

Benutzer: *vmuser*
Passwort: *codypp-1819*

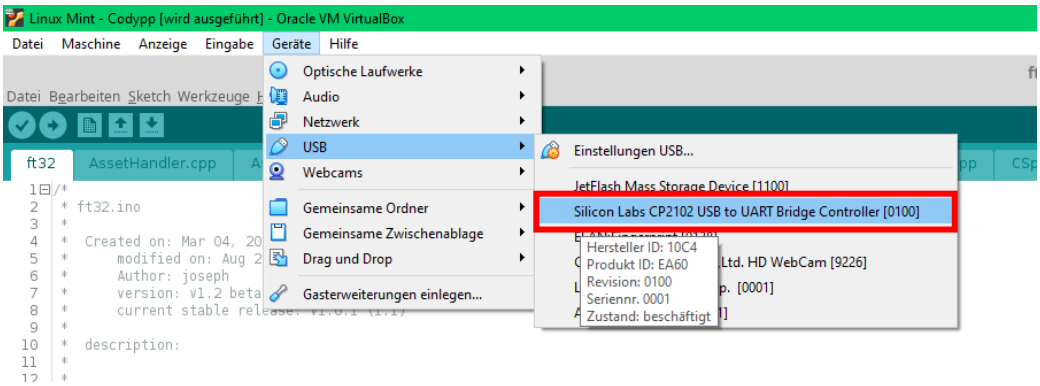


Starten Sie nach dem Einloggen die Arduino IDE (blaues Arduino Logo). Verbinden Sie danach den ESP32 über die Serielle Schnittstelle (USB) mit dem PC.

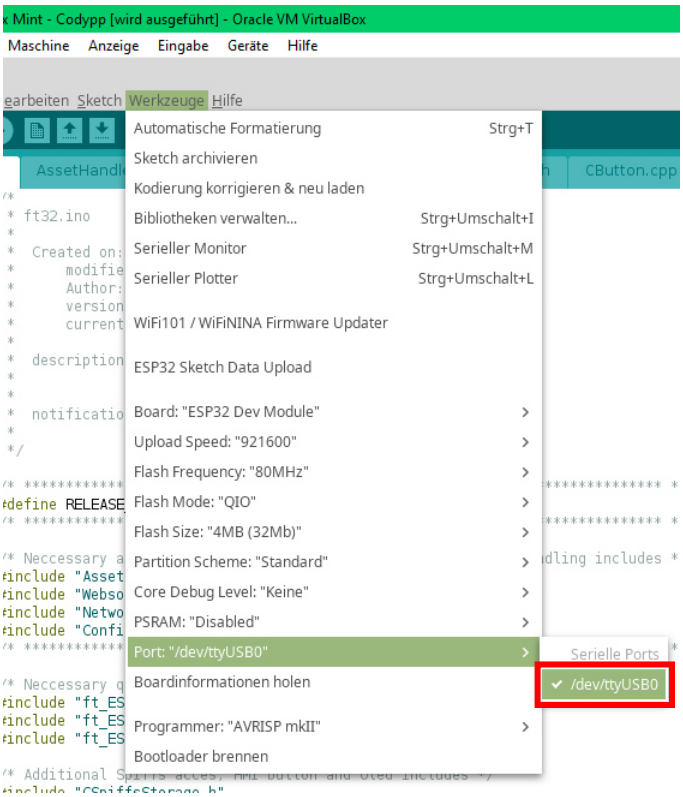


arduino
Anwendung

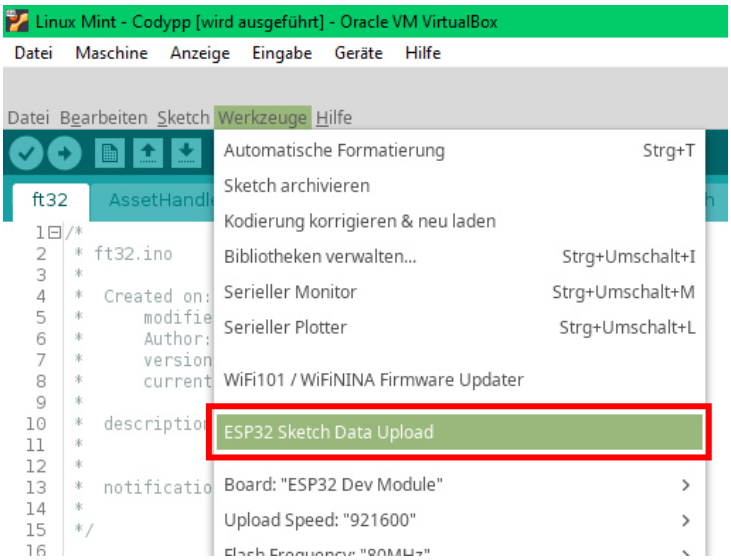
Wählen Sie unter *„Geräte -> USB“* in der Menüleiste der Virtuellen Maschine den *„Silicon Labs CP2102 USB to UART Bridge Controller“* aus.



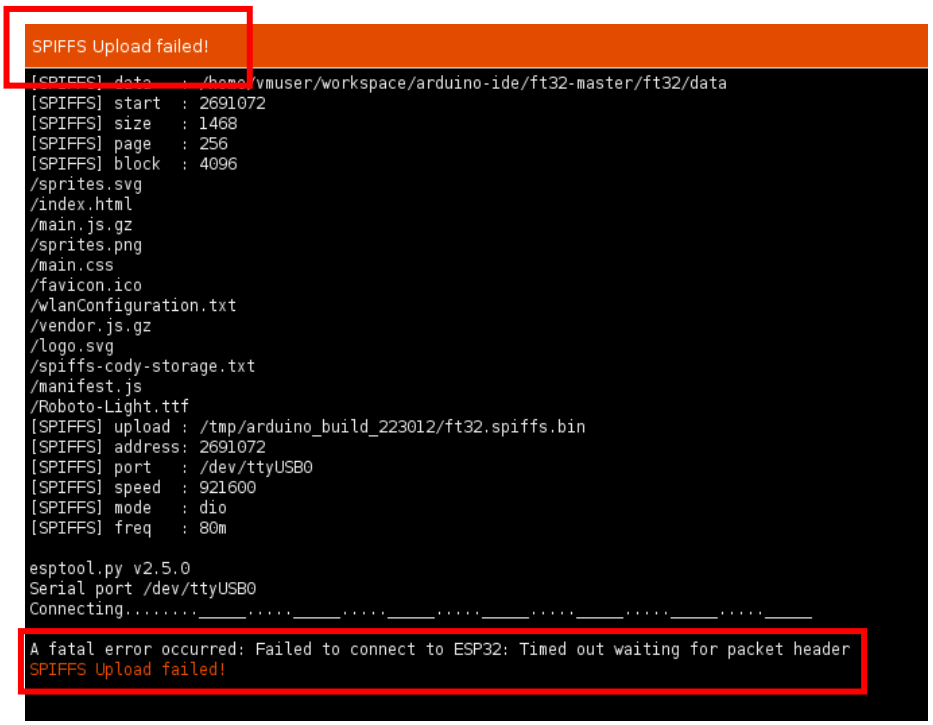
Unter *„Werkzeuge -> Port“* kann die korrekte Verbindung überprüft werden.
Es sollte die Verbindung mit *„/dev/ttyUSB0“* angezeigt werden.



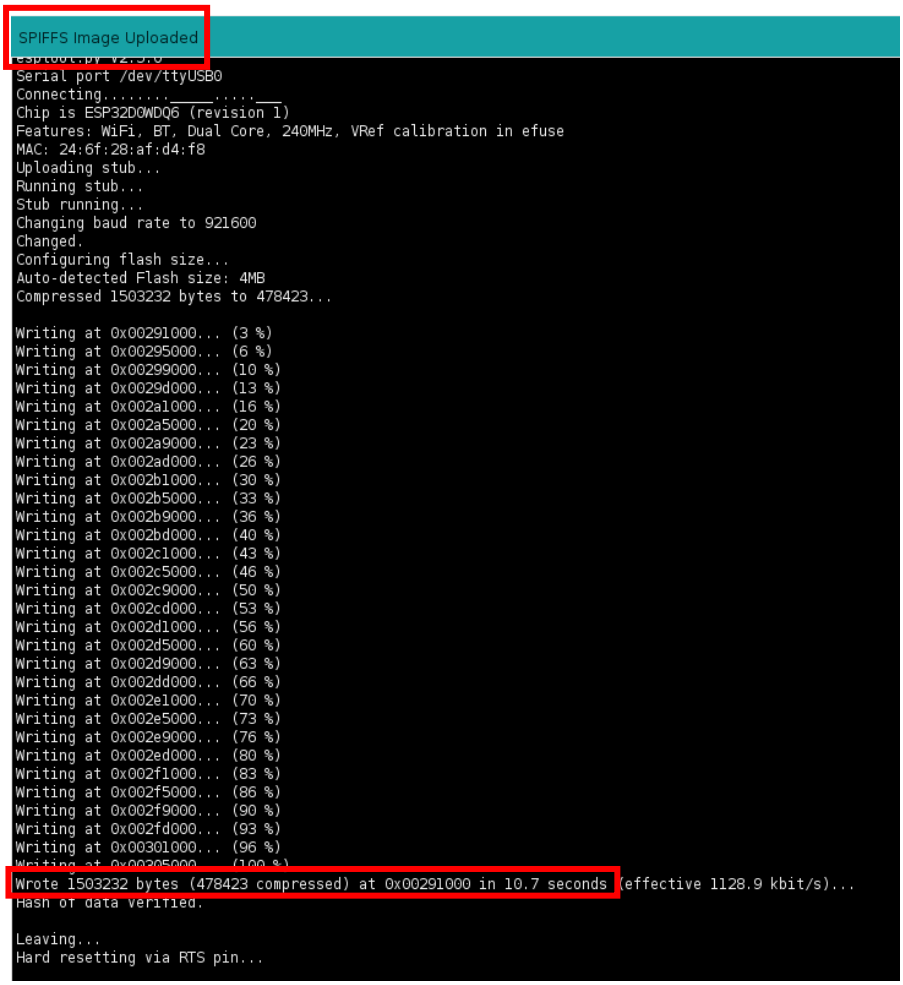
Unter *„Werkzeuge -> ESP32 Sketch Data Upload“* können die SPIFFS beschrieben werden.



Hinweis: Falls beim Beschreiben der SPIFFS ein Fehler auftritt, (siehe nachfolgendes Bild) muss der „BOOT“ Button auf dem ESP32-Development-Board gedrückt und gehalten werden bis weitere Konsolenausgaben auftreten.

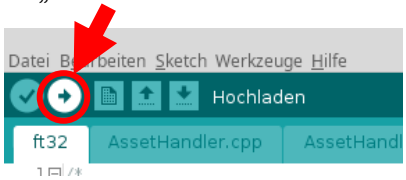


Nach erneutem Betätigen des *„Werkzeuge -> ESP32 Sketch Data Upload“* Button sowie dem Halten des „BOOT“ Buttons sollte der erfolgreiche Verbindungsaufbau mit dem ESP32 und das erfolgreiche Hochladen der SPIFFS mit folgender Meldung in der Konsole zu sehen sein.

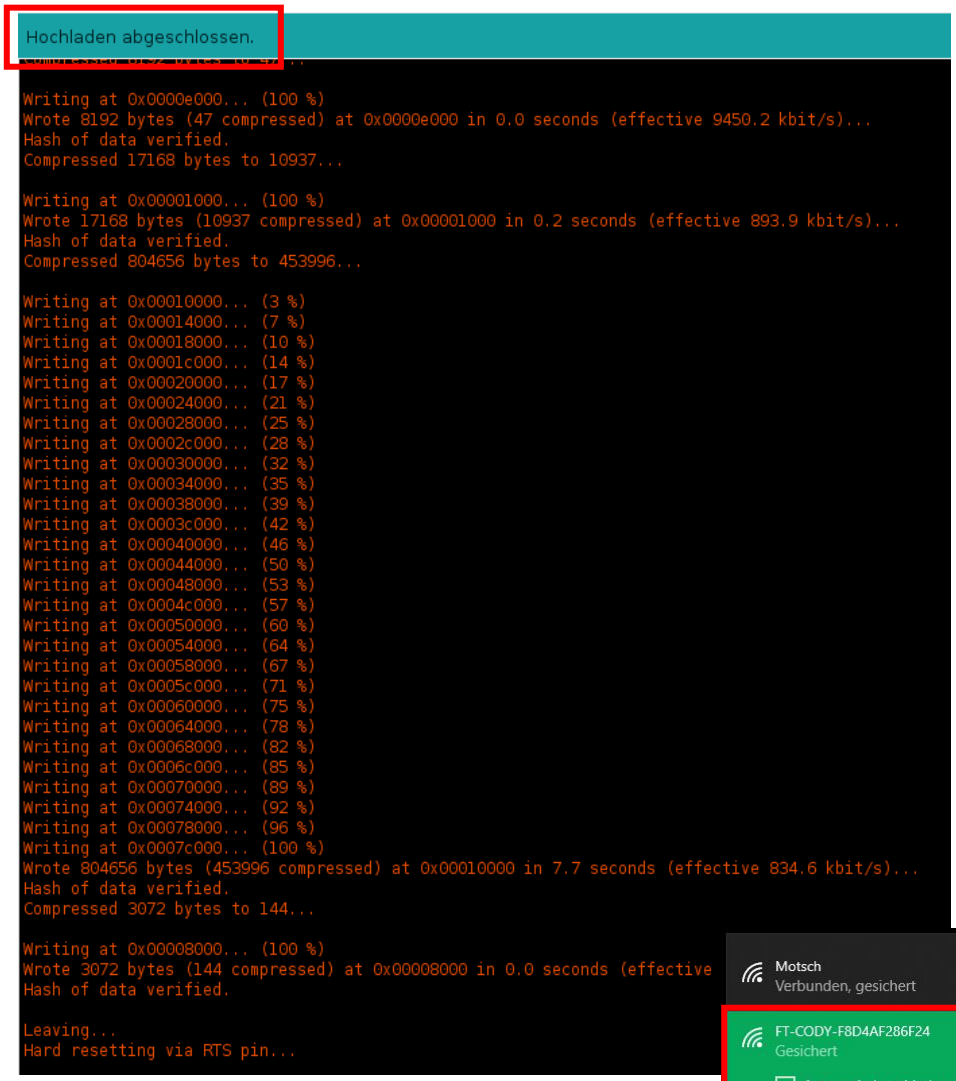


Hinweis: Die hochzuladenden SPIFF Dateien befinden sich im Linux Ordner unter: *“/home/vmuser/workspace/arduino-ide/ft32-master/ft32/data”*

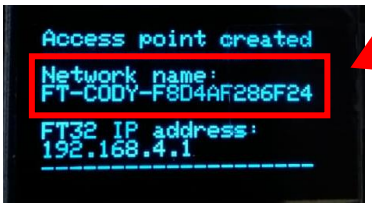
Laden Sie das „Backend“ mit dem *„Hochladen“* Button in der Menüleiste der Arduino IDE hoch.

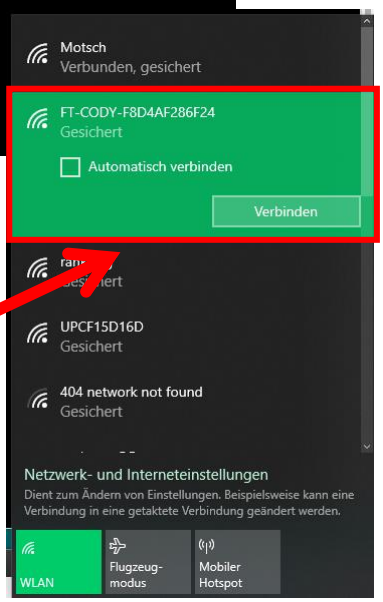


Sollte der gleiche Fehler wie beim Beschreiben der SPIFFS auftreten, müssen Sie mit erneuter Betätigung der „Boot“ Taste den Vorgang wiederholen (siehe oben). Bei erfolgreichem „Flashen“ des „Backends“ erscheint folgende Meldung in der Konsole:

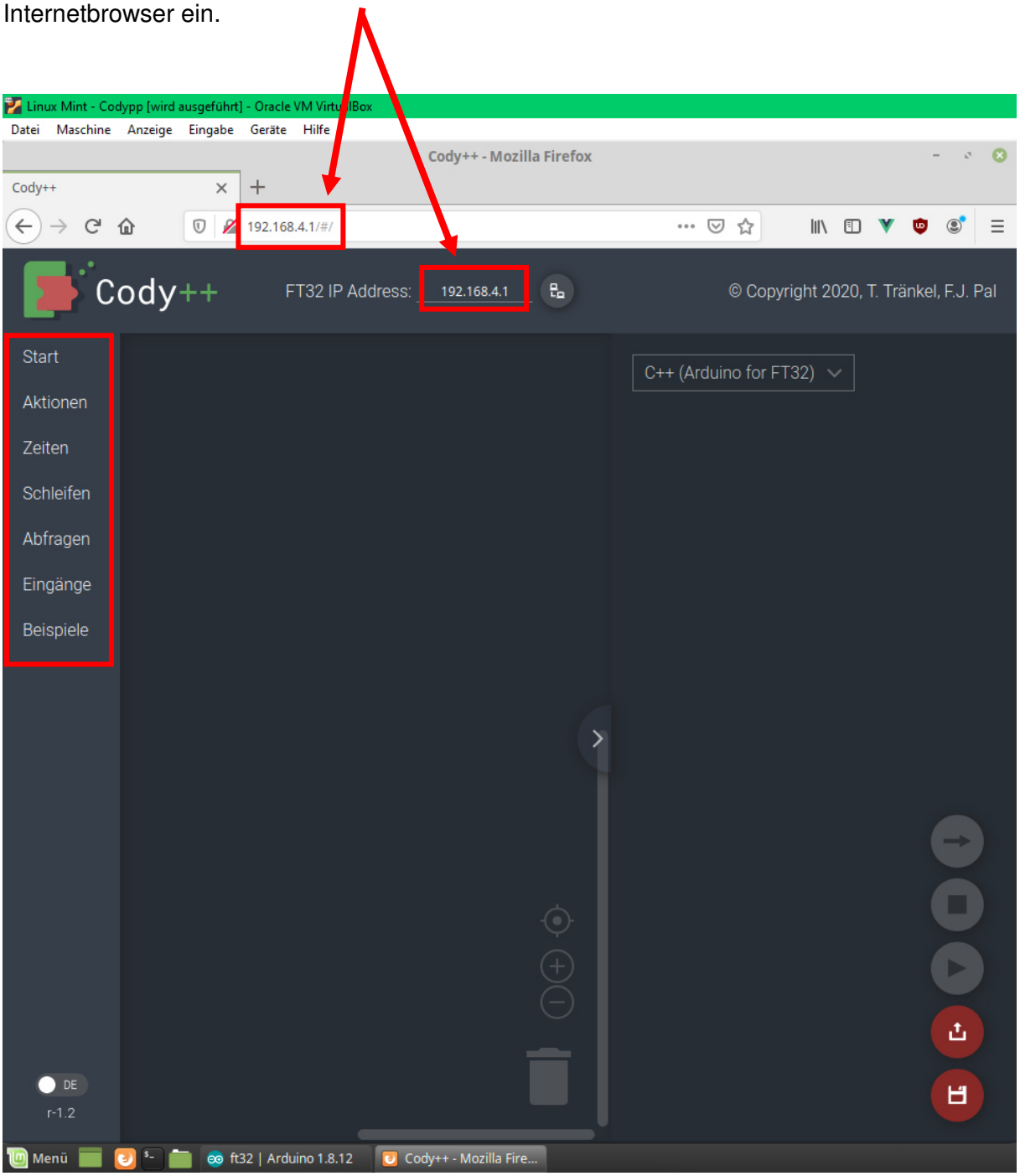


Bitte verbinden Sie Ihren PC nun mit dem Netzwerk des ESP32 eMalRob-FG, dessen Netzwerk Name auf dem OLED Display, welcher auf dem Roboter angebracht ist, erscheint.





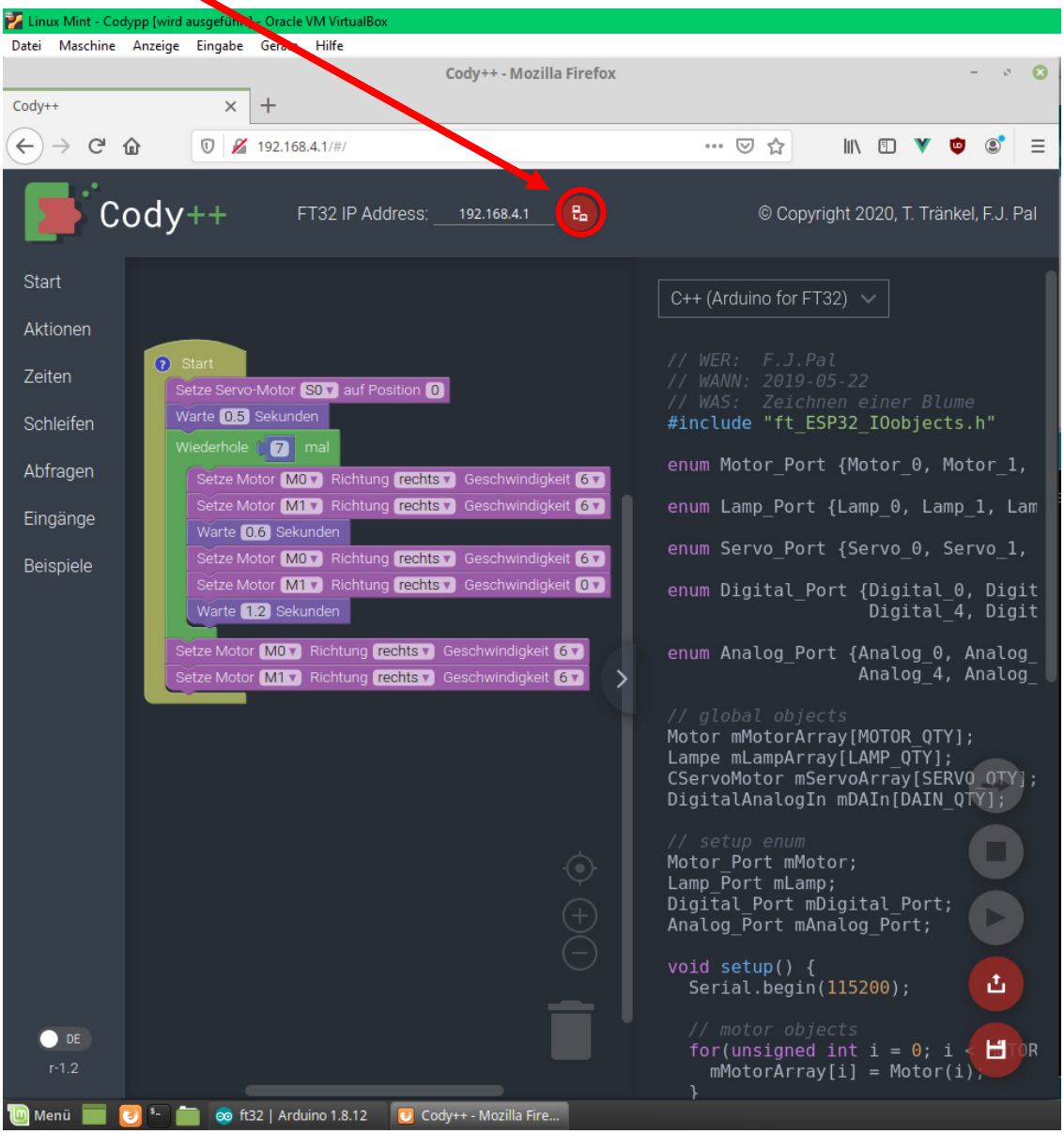
Geben Sie nun die IP-Adresse, die auf dem OLED Display angezeigt wird in Ihrem Internetbrowser ein.



Nun befinden Sie sich in der sogenannten Cody++ Programmieroberfläche mit direktem Zugang zum ESP32. Auf der linken Seite sehen Sie verschiedene Überbegriffe für Programmierbausteine. Wählen Sie einen gewünschten Überbegriff aus und ziehen Sie einen Unterbaustein anhand Drag & Drop ins Programmierfeld.

Hinweis: Jeder Programmierung muss mit dem „Start“ Block begonnen werden.
Dieser Block fasst den kompletten Programmcode ein und dient als Initialisierung und Setup.

In folgendem Bild sehen Sie ein Beispiel/Muster Programm. Haben Sie das Programmieren abgeschlossen, öffnen Sie die Datenübertragung indem Sie oben in der Mitte auf das *„Verbinden“* Symbol klicken.



Ist eine Verbindung aufgebaut, kann über den roten „Pfeil“ Button das Programm an den ESP gesendet werden. Mit dem „Play“ Button wird das Programm anschließend auf dem eMalRob-FG gestartet. Jetzt müsste sich der Roboter je nach Programmierung bewegen.

Hinweis: Stoppen des Roboters wird über die „Stopp“ Button, Pausieren während des Programmablaufs wird über den „Pause“ Button realisiert. Zusätzlich kann das selbst erstellte Programm gespeichert oder ein bestehendes Programm hochgeladen werden.

S2 – Cody++ Programmierung extern auf dem Rechner

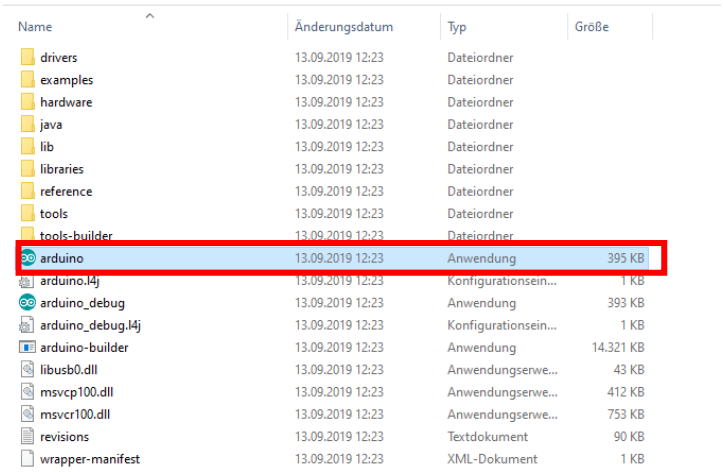
2.1 – via Serieller Schnittstelle (USB)

Der erste Schritt ist das Downloaden des Zip Ordners auf GitHub. Dieser Ordner dient als komplette Zusammenfassung aller benötigten Dateien.

Zip Dateien von GitHub sind unter folgendem Link erhältlich:
https://github.com/Mechatronikwelt/eMalRob-FG.git

Danach müssen Sie auf Ihrem PC die Arduino Software IDE (integrierte Entwicklungsumgebung) installieren. Die Arduino Software kann kostenlos auf der offiziellen Homepage von Arduino heruntergeladen werden. Installieren Sie die **Version 1.8.10**.
https://www.arduino.cc/en/Main/OldSoftwareReleases#previous

Sobald sie den Zip Ordner heruntergeladen und entpackt haben, können Sie die darin enthaltene „Arduino.exe“ ausführen. Somit startet sich das Programm.



Name	Änderungsdatum	Typ	Größe
drivers	13.09.2019 12:23	Dateiordner	
examples	13.09.2019 12:23	Dateiordner	
hardware	13.09.2019 12:23	Dateiordner	
java	13.09.2019 12:23	Dateiordner	
lib	13.09.2019 12:23	Dateiordner	
libraries	13.09.2019 12:23	Dateiordner	
reference	13.09.2019 12:23	Dateiordner	
tools	13.09.2019 12:23	Dateiordner	
tools-builder	13.09.2019 12:23	Dateiordner	
arduino	13.09.2019 12:23	Anwendung	395 KB
arduino.l4j	13.09.2019 12:23	Konfigurationsein...	1 KB
arduino_debug	13.09.2019 12:23	Anwendung	393 KB
arduino_debug.l4j	13.09.2019 12:23	Konfigurationsein...	1 KB
arduino-builder	13.09.2019 12:23	Anwendung	14.321 KB
libusb0.dll	13.09.2019 12:23	Anwendungserwe...	43 KB
msvcp100.dll	13.09.2019 12:23	Anwendungserwe...	412 KB
msvcr100.dll	13.09.2019 12:23	Anwendungserwe...	753 KB
revisions	13.09.2019 12:23	Textdokument	90 KB
wrapper-manifest	13.09.2019 12:23	XML-Dokument	1 KB

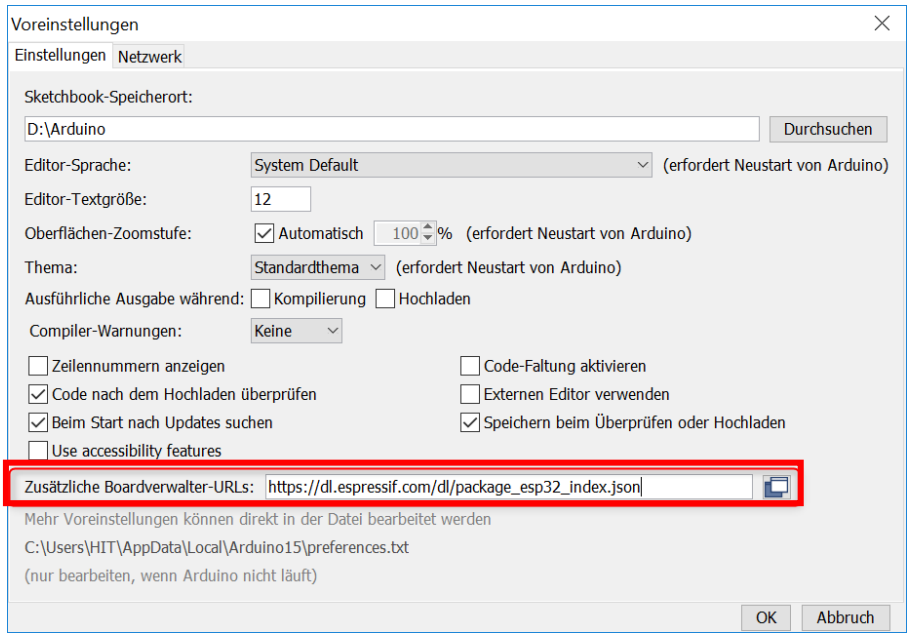
Nun öffnet sich die Arduino Software. In der Leiste links oben können Sie überprüfen, ob die korrekte Version ausgeführt wird.

Damit eine korrekte Verbindung zum eMalRob-FG hergestellt werden kann, muss der Boardverwalter noch entsprechend eingestellt werden. Hierzu müssen Sie folgende Schritte ausführen:

Kopieren Sie den sogenannten „JSON“ Link:

https://dl.espressif.com/dl/package_esp32_index.json

und fügen ihn unter *„Datei ->Voreinstellungen ->Zusätzliche Boardverwalter URLs“* ein. Bestätigen Sie anschließend mit *„OK“*.

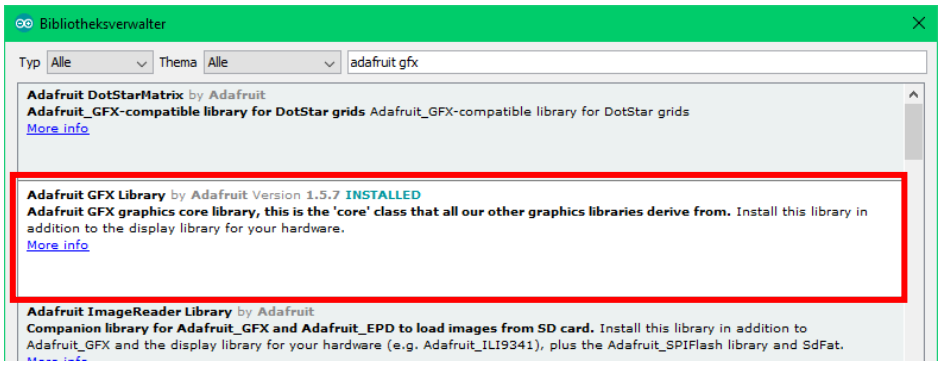


Damit eine korrekte Verbindung zum eMalRob-FG hergestellt werden kann, müssen bestimmte, notwendige Bibliotheken in die Arduino Software eingebunden werden. Folgende Bibliotheken müssen Sie nun einbinden:

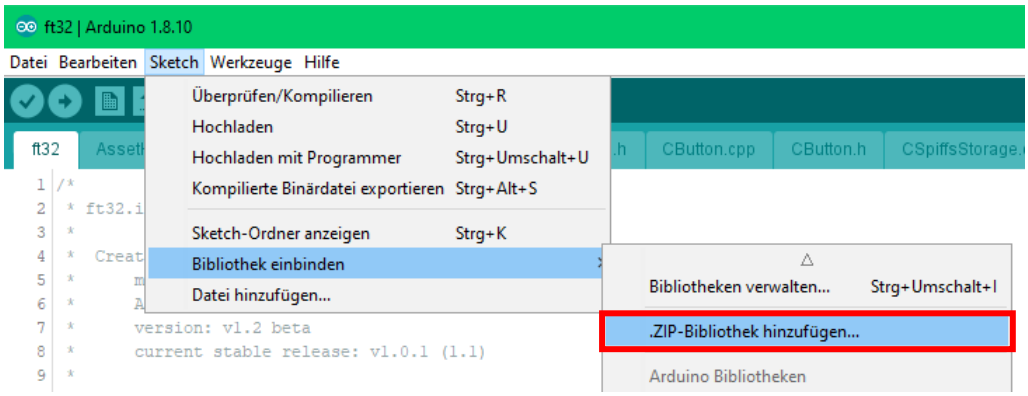
1) *Adafruit GFX Library by Adafruit* ***Version 1.5.7***
2) *Adafruit SSD1306 by Adafruit* ***Version 1.3.0***
3) *ESP32Servo by John K. Bennett, Kevin Harrington* ***Version 0.6.0***

Wählen Sie im Reiter *„Werkzeuge“* das Feld *„Bibliotheken verwalten…“*. Es öffnet sich wiederum ein Fenster. Suchen Sie über das Suchfeld rechts oben die erforderlichen Bibliotheken und installieren diese.

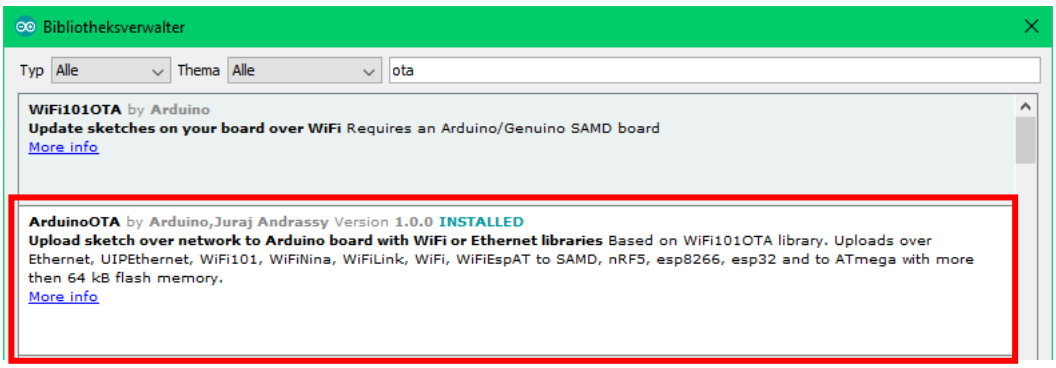
Hinweis: Achten Sie darauf, die richtige Version der Bibliotheken zu installieren.



Anschließend müssen weitere Bibliotheken eingebunden werden. (Zu finden im GitHub Zip Ordner). Wählen Sie hierzu in der Arduino IDE im Reiter *„Sketch“* das Feld *„Bibliothek einbinden“* und wählen daraufhin *„.ZIP-Bibliothek hinzufügen…“.* Suchen Sie nun die „ESP32MotorControl-master.zip“ über das Explorer-Fenster und drücken Sie öffnen. Wiederholen Sie diesen Schritt auch für die „MalRob_FG.zip“.



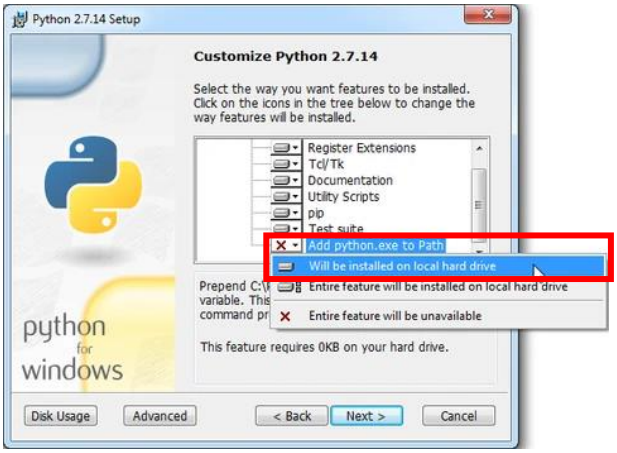
Jetzt wird die Bibliothek „Arduino OTA.h“ implementiert. Hierzu wählen Sie in der Arduino IDE im Reiter *„Sketch“* das Feld *„Bibliothek einbinden“* und wählen daraufhin *„Bibliotheken verwalten…“.* Suchen Sie nun nach der „ArduinoOTA“ unter dem Suchbegriff „OTA“ über das Explorer-Fenster und drücken Sie *„installieren“*. Installieren Sie die **Version 1.0.0**.



Danach installieren Sie die vorhandene Python *.msi* Datei „python-2.7.18.amd64“ (Zu finden im GitHub Zip Ordner).

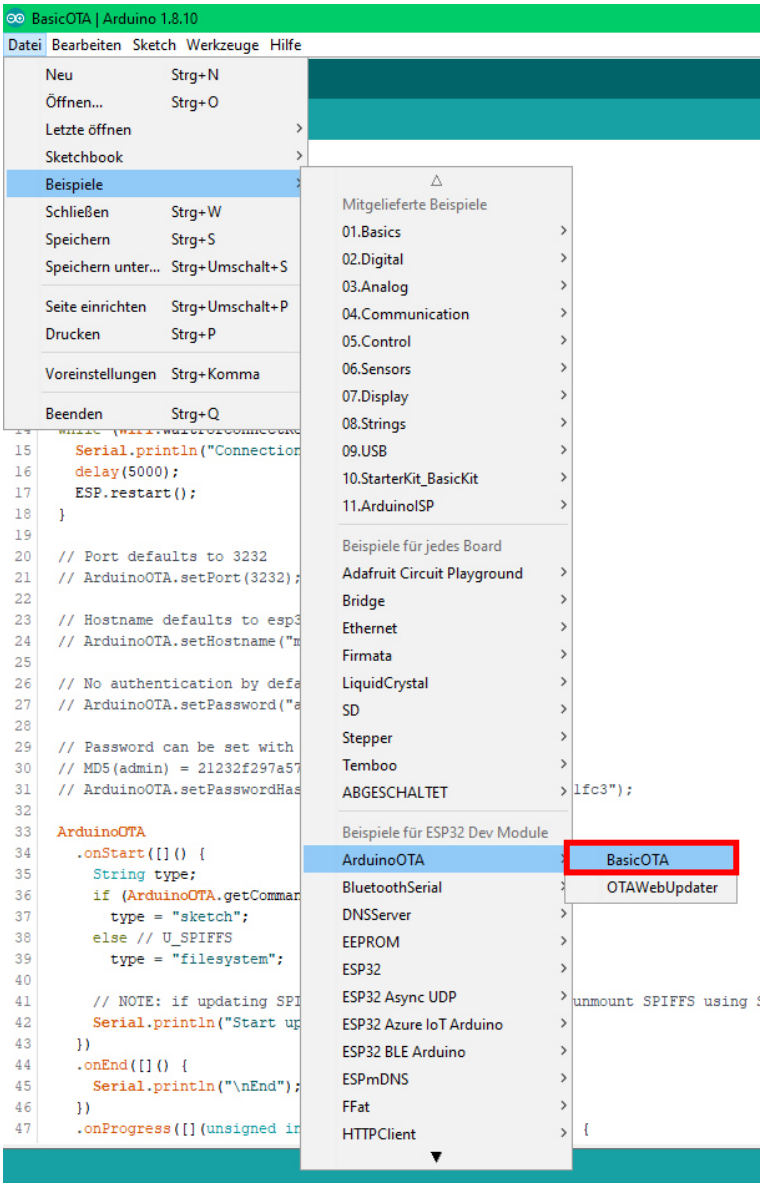
Führen Sie die „msi“ Datei aus und folgen den Installationsanweisungen.

Hinweis: Bitte klicken Sie zusätzlich auf *„Add python.exe to Path“* und wählen Sie die Option *„Will be installed on local hard drive“* aus.



Anschließend starten Sie Ihren PC neu.

Um später den „OTA“ Vorgang benutzen zu können, müssen Sie zuerst den dazu benötigten Programmcode unter *„Datei -> Beispiele ->ArduinoOTA -> BasicOTA“* aufrufen.



Sofern Sie den Code sehen, tragen Sie bitte die Ihnen zur Verfügung stehenden WLAN-Informationen des Internet-Routers ein. Die SSID ist der Name des Routers/Zugangspunkts und das Passwort ist der WPA Netzwerkschlüssel.

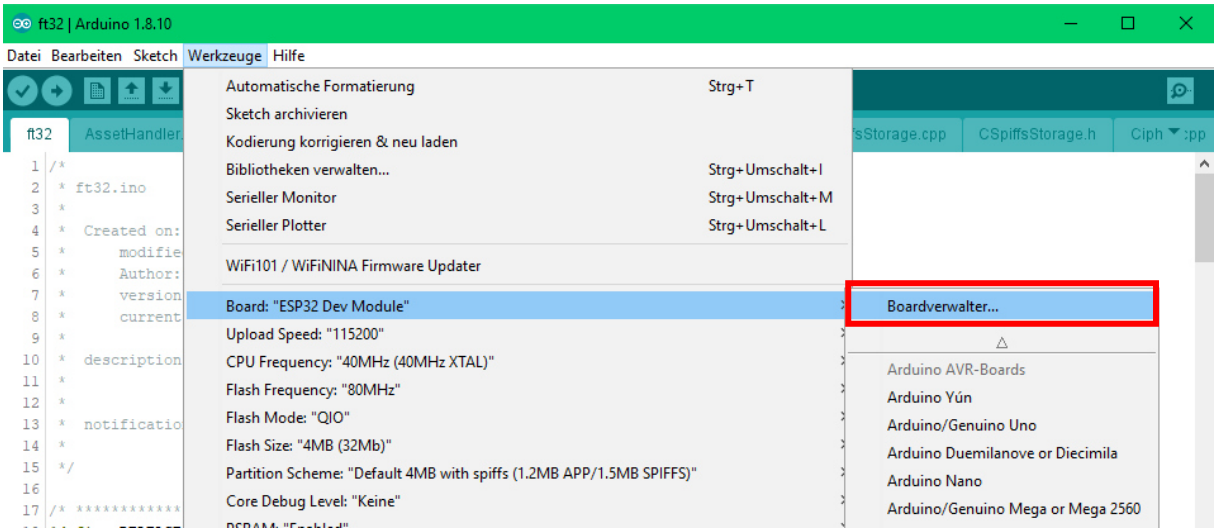
Gleich am Anfang müssen wir unsere WLAN-Informationen eintragen:

```
const char* ssid = ".........";
const char* password = ".........";
```

Diesen ganzen Code müssen wir in Zukunft mit in unser Programm einbauen und ganz wichtig die Zeile ArduinoOTA.handle(); nicht enfernen. Sollte das passiert sein, muss der nächste Programmiervorgang wieder mit Programmer über die Serielle Schnittstelle geschehen.

```
void loop() {
  ArduinoOTA.handle();
}
```

Nun müssen Sie die Boardeinstellungen anpassen. Dies funktioniert unter: *„Werkzeuge -> Board -> Boardverwalter“*

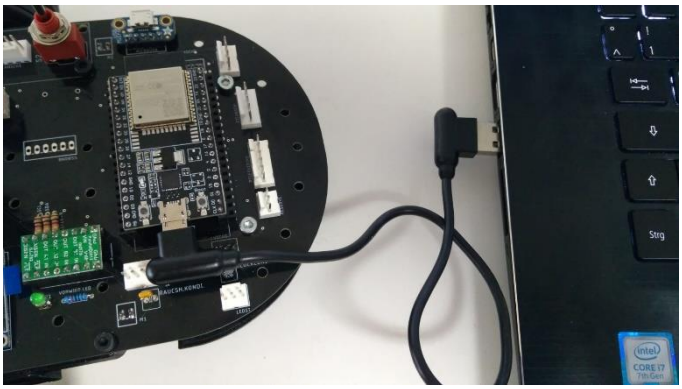


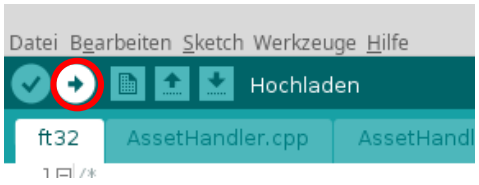
Anschließend suchen Sie mit dem Begriff „ESP“ unter *„Werkzeuge -> Board -> Boardverwalter“* nach „esp32 by Espressif Systems“ und installieren Sie die **Version 1.0.4.**

Ändern Sie anschließend die Parameter im Reiter *„Werkzeuge“* auf folgende Einstellungen ab:

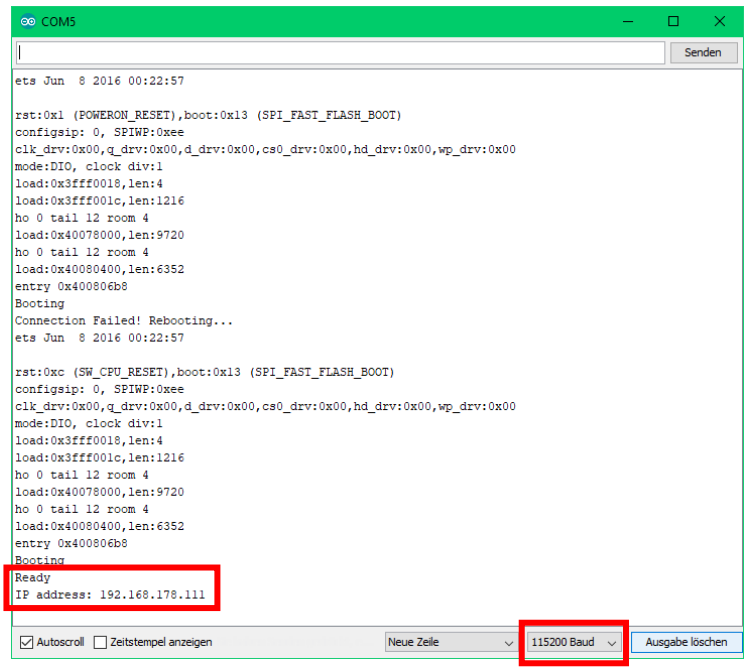
→ Board:	ESP32 Dev Module
→ Upload Speed:	115200
→ CPU Frequency:	240MHz
→ Flash Frequenzy:	80MHz
→ Flash Mode:	QIO
→ Flash Size:	4MB
→ Partition Scheme:	Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)
→ Core Debug Level:	Keine
→ PSRAM:	Enabled

Nun können Sie via serieller Schnittstelle (USB) den „BasicOTA“ Programmcode, welchen Sie mir Ihrem Netzwerk Daten versehen haben, auf den ESP32 „flashen“. Dies funktioniert mit bestehender Verbindung mit dem PC sowie des Betätigen des *„Hochladen“* Buttons.





Öffnen Sie nach erfolgreichem Hochladen den Seriellen Monitor unter *„Werkzeuge -> Serieller Monitor“* und stellen Sie eine Bautrate von *„115200“* (untere Leiste) ein. Ebenso drücken Sie direkt im Anschluss den „EN“ Button auf dem ESP32 Development Board.
Bei erfolgreichen Laden wird Ihnen die dynamische IP Adresse, die der Router vergeben hat, angezeigt.
Bitte notieren Sie diese.
(Hier als Beispiel: 192.168.178.111)

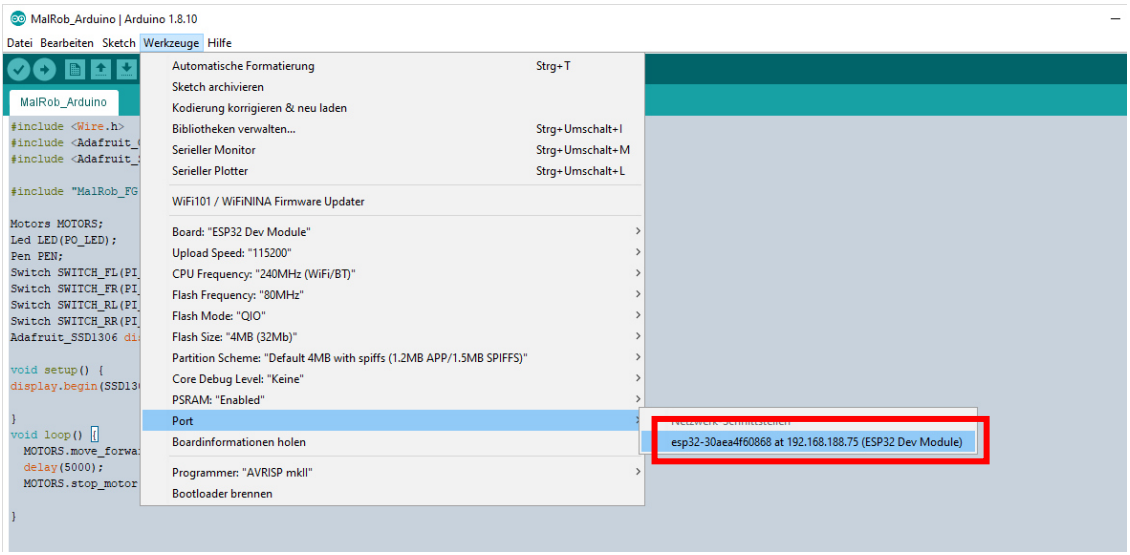


Hinweis: Sofern unter *„Werkzeuge -> Port“* ein „COM-Port“ ausgewählt ist, wird die Programmcode-übertragung seriell (via USB) erfolgen. Die Zahl nach „COM“ kann von eMalRob-FG und PC variieren. Für eine OTA (Over the air) Übertragung befolgen Sie die weiteren Schritte.

2.2 – via WIFI

Nach dem Übertragen via serieller Schnittstelle (USB), können Sie Ihren erstellten Bewegungsablauf in den „BasicOTA“ Code einbinden. Achten Sie auf korrekte Einbindung der verschiedenen Codezeilen im Implementierungs-, Setup- und Loop-Bereich! Die serielle Verbindung des ESP zum PC kann nun getrennt werden. Eine Spannungsversorgung des kompletten eMalRob-FG, sollte mit der beiliegenden Powerbank und dem Micro-USB-Port außerhalb des ESP32 gewährleistet sein.

Anschließend wählen Sie unter *„Werkzeuge -> Port“* die angezeigte ESP/Netzwerk-Schnittstelle mit dazugehöriger IP Adresse aus.



Nun kann das Programm wie gewohnt mit dem „Pfeil“ Button auf den eMalRob-FG hochgeladen werden.

Folgend wird ein kurzer Beispiel-/Muster Code zu Implementierung gezeigt, welcher so auf dem eMalRob-FG genutzt werden kann. Der Code befindet sich ebenfalls im GitHub Zip Ordner.

```
//Includes für OTA
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>

//Includes für eMalRob-FG
#include "Ticker.h"
#include <arduino.h>
#include <analogWrite.h>
#define AIN1 17        //Pindefinitionen für ESP32
#define AIN2 16
#define BIN1 4
#define BIN2 2
#define EN 27
#define Pin 39
#define Pin1 36
int s=0;
int var;
int s1=0;
int var1;
int Z1=0;
int Z=0;
int Y1=0;
int Y=0;

//WLAN-ROUTER-DATEN
const char* ssid = "..WLAN-ROUTER-NAME..";
const char* password = "..WLAN-ROUTER-PASSWORT..";

void setup() {

  /** BEGIN OTA SETUP **/

  Serial.begin(115200);
  Serial.println("Booting");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connection Failed! Rebooting...");
    delay(5000);
    ESP.restart();
  }

  // Port defaults to 3232
  // ArduinoOTA.setPort(3232);
  // Hostname defaults to esp3232-[MAC]
  // ArduinoOTA.setHostname("myesp32");
  // No authentication by default
  // ArduinoOTA.setPassword("admin");
  // Password can be set with it's md5 value as well
  // MD5(admin) = 21232f297a57a5a743894a0e4a801fc3
  // ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3");

  ArduinoOTA
    .onStart([]() {
      String type;
      if (ArduinoOTA.getCommand() == U_FLASH)
        type = "sketch";
      else // U_SPIFFS
        type = "filesystem";

      // NOTE: if updating SPIFFS this would be the place to unmount SPIFFS using SPIFFS.end()
      Serial.println("Start updating " + type);
    })
    .onEnd([]() {
      Serial.println("\nEnd");
    })
    .onProgress([](unsigned int progress, unsigned int total) {
      Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
    })
```

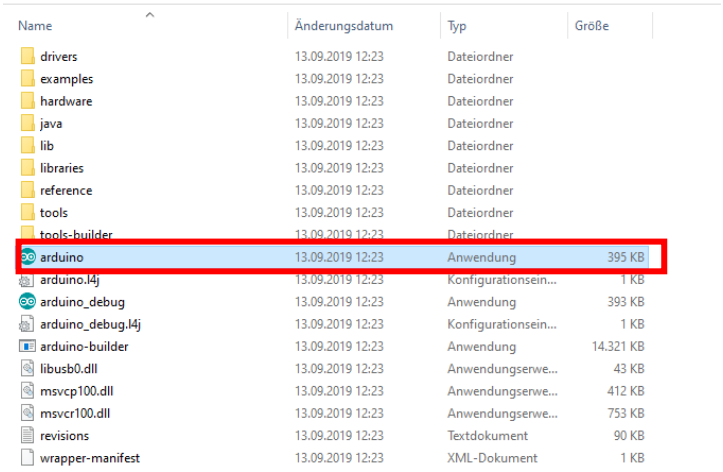
```
    .onError([](ota_error_t error) {
      Serial.printf("Error[%u]: ", error);
      if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
      else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
      else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
      else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
      else if (error == OTA_END_ERROR) Serial.println("End Failed");
    });

  ArduinoOTA.begin();

  Serial.println("Ready");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  /** OTA SETUP END **/

  /** BEGIN MOTOR-PINS SETUP **/

  pinMode(AIN1,OUTPUT);
  pinMode(AIN2,OUTPUT);
  pinMode(BIN1,OUTPUT);
  pinMode(BIN2,OUTPUT);
  pinMode(EN,OUTPUT);
  pinMode(Pin,INPUT);
  pinMode(Pin1,INPUT);
  digitalWrite(EN,HIGH);

  /** MOTOR-PINS SETUP END **/
}


void loop() {
  ArduinoOTA.handle();

  digitalWrite(EN,HIGH);     // Einschalten des Motortreibers
  analogWrite(AIN1,255);     //Rechts
  analogWrite(AIN2,HIGH);    //fährt rückwärts/ fast decay (Motor 1)

  analogWrite(BIN1,255);     //Links
  analogWrite(BIN2,HIGH);    //fährt vorwärts/ fast decay (Motor 2)

  delay(1000);
  analogWrite(BIN1,HIGH);
  analogWrite(BIN2,HIGH);
  analogWrite(AIN1,LOW);
  analogWrite(AIN2,LOW);
  delay(1000);

  analogWrite(BIN1,LOW);
  analogWrite(BIN2,LOW);
  analogWrite(AIN1,LOW);
  analogWrite(AIN2,LOW);
  digitalWrite(EN,LOW);     // Auschalten des Motortreibers
  delay(1000);
}
```

S3 – Fernsteuerung des eMalRob via Smartphone-App

Der erste Schritt ist das Downloaden des Zip Ordners auf GitHub. Dieser Ordner dient als komplette Zusammenfassung aller benötigten Dateien.
Zip Dateien von GitHub sind unter folgendem Link erhältlich:
https://github.com/Mechatronikwelt/eMalRob-FG.git

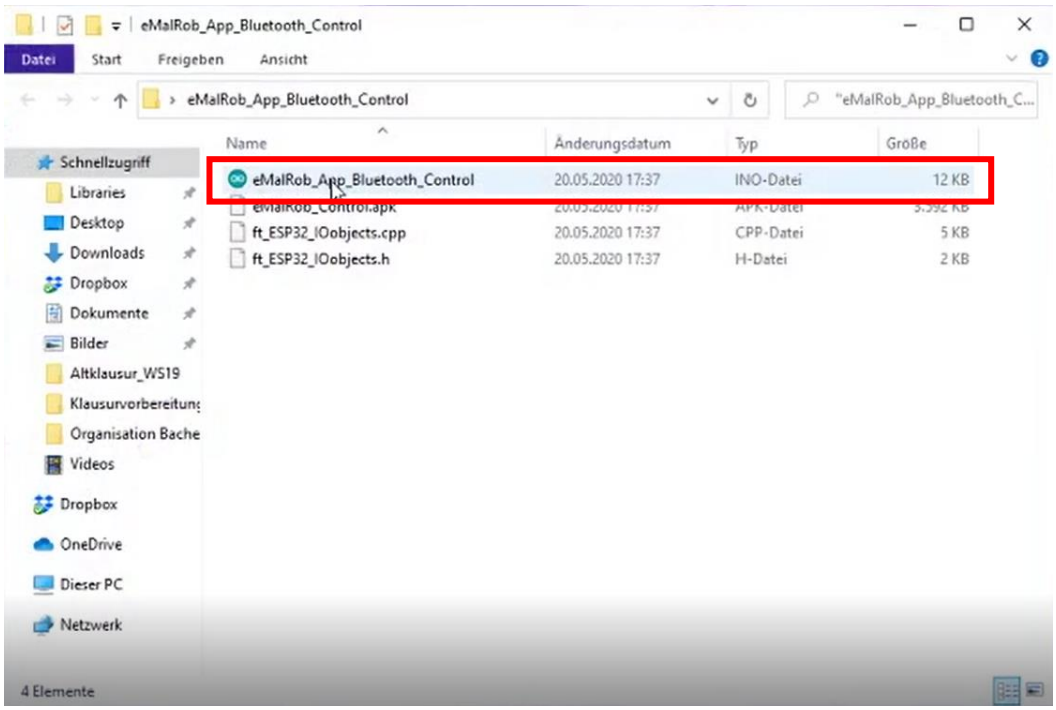
Sofern nicht vorhanden, müssen Sie auf Ihrem PC die Arduino Software IDE (integrierte Entwicklungsumgebung) installieren. Die Arduino Software kann kostenlos auf der offiziellen Homepage von Arduino heruntergeladen werden. Installieren Sie die **Version 1.8.10**.
https://www.arduino.cc/en/Main/OldSoftwareReleases#previous

Sobald sie den Zip Ordner heruntergeladen und entpackt haben, können Sie die darin enthaltene „Arduino.exe“ ausführen. Somit startet sich das Programm.



Nun öffnet sich die Arduino Software. In der Leiste links oben können Sie überprüfen, ob die korrekte Version ausgeführt wird.

Bitte öffnen Sie die Datei „eMalRob_App_Bluetooth_Control.ino“ aus dem entpackten „eMalRob_App_Bluetooth_Control“ Zip Ordner (zu finden im GitHub Zip Ordner) über *„Datei -> Öffnen“* in ihrem Arduino Programm.



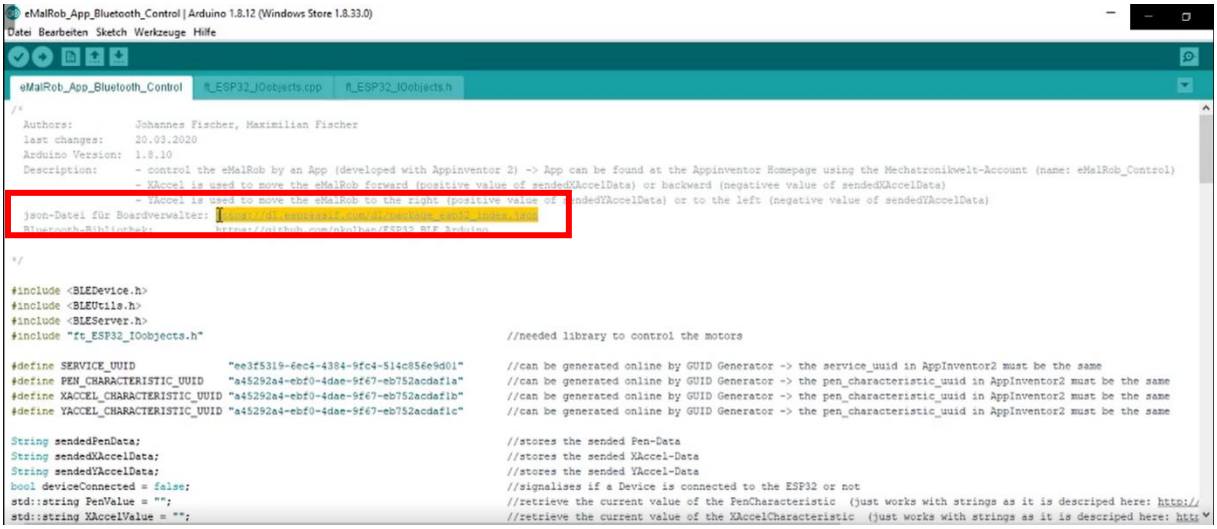
Damit eine korrekte Verbindung zum eMalRob-FG hergestellt werden kann, muss der Boardverwalter noch entsprechend eingestellt werden. Hierzu müssen Sie folgende Schritte ausführen:

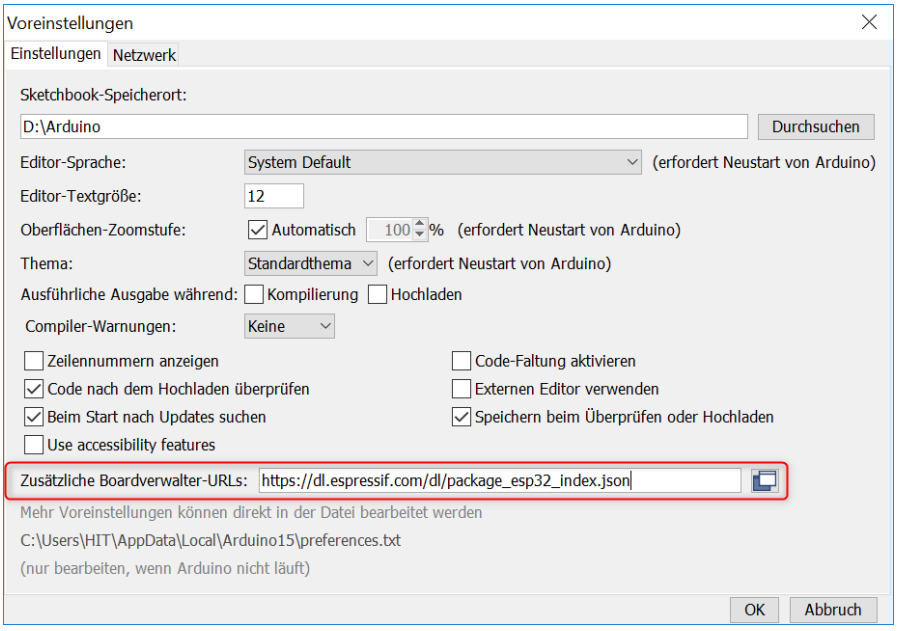
Kopieren Sie den sogenannten „JSON“ Link:

https://dl.espressif.com/dl/package_esp32_index.json

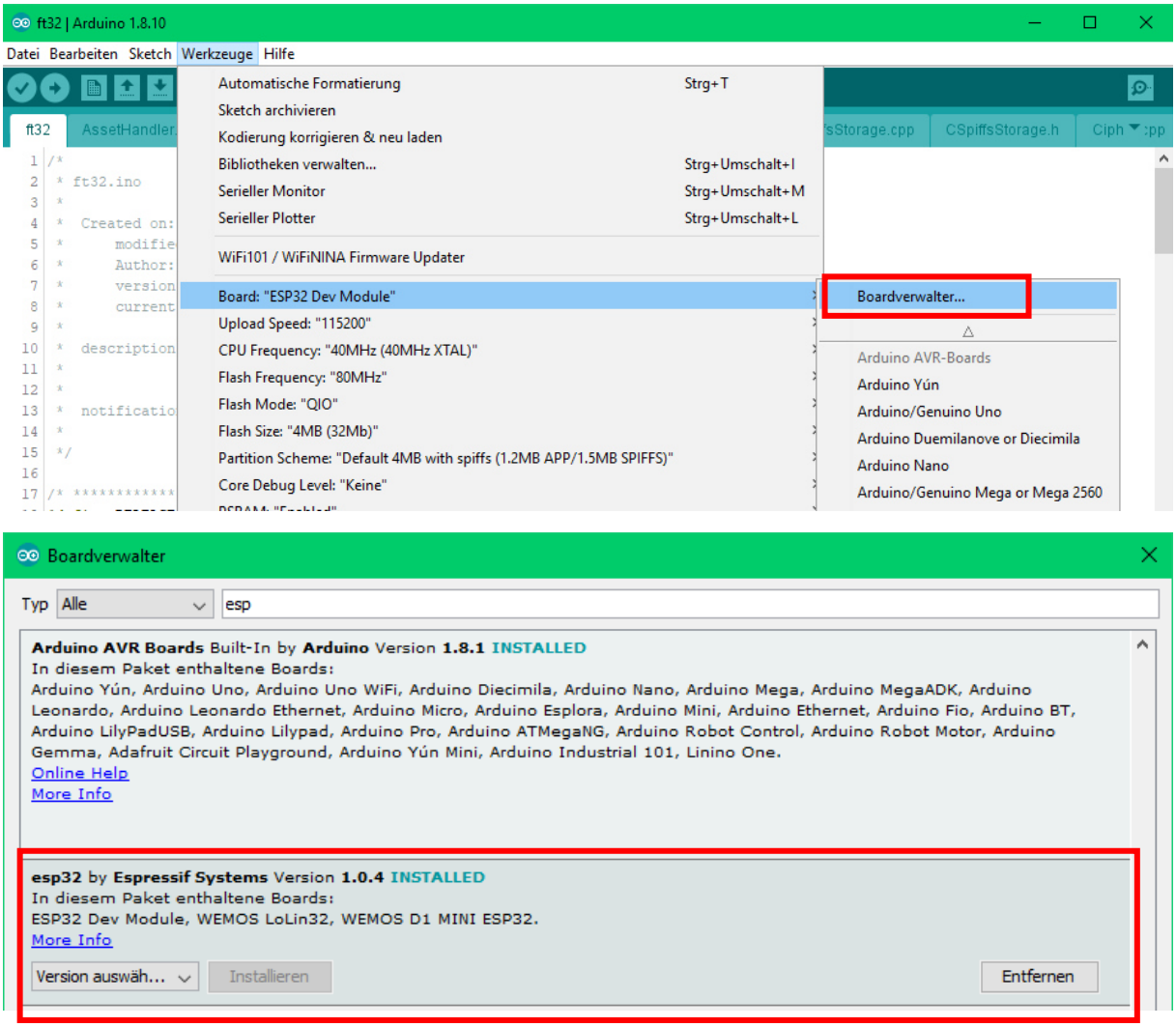
und fügen ihn unter *„Datei ->Voreinstellungen ->Zusätzliche Boardverwalter URLs“* ein. Bestätigen Sie anschließend mit *„OK“*.

Hinweis: Der „JSON“ Link befindet ebenso im oberen Kommentarfeld des Programmcodes.



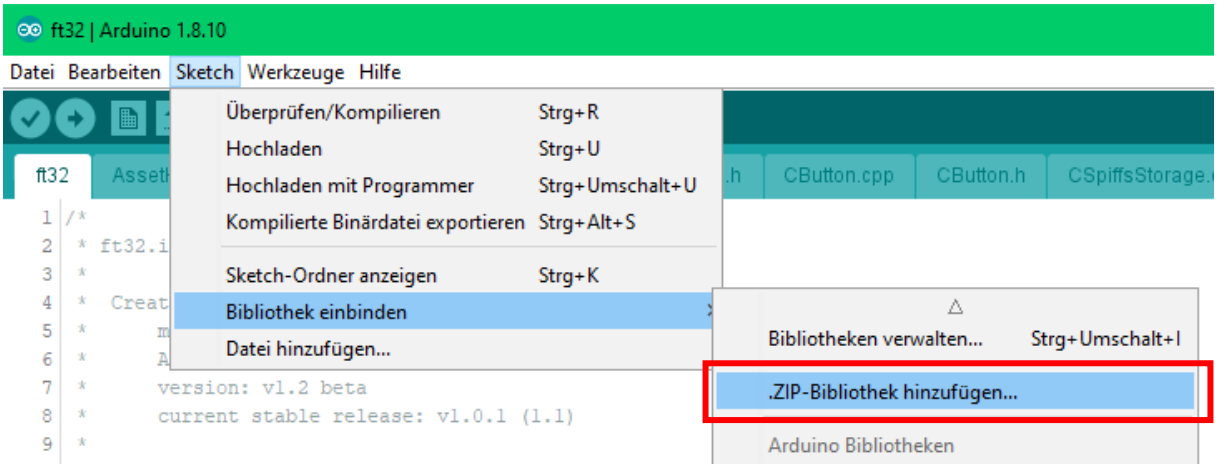


Anschließend suchen Sie mit dem Begriff „ESP“ unter *„Werkzeuge -> Board -> Boardverwalter“* nach „esp32 by Espressif Systems“ und installieren Sie die **Version 1.0.4**.

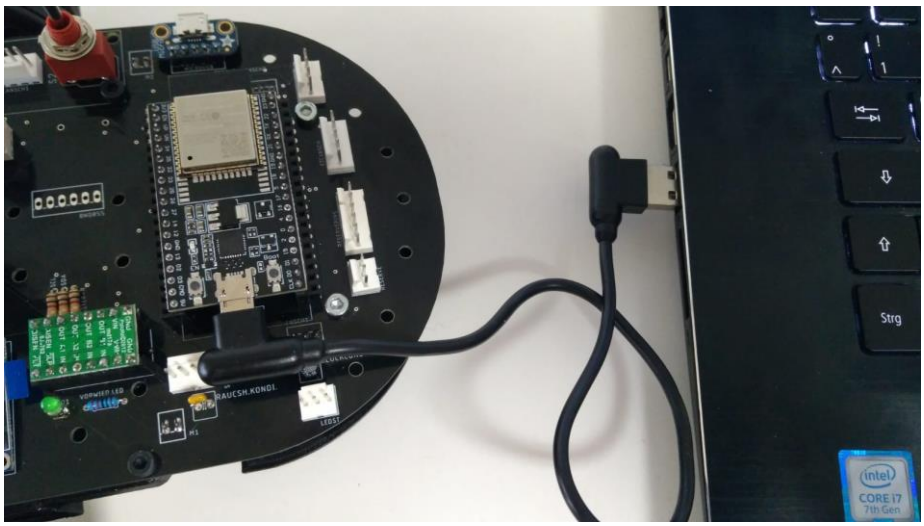


Nun sollte bei Ihnen unter *„Werkzeuge -> Board -> ESPDevModul“* der ESP32 mit „ESP32 Dev Modul“ Kennung erscheinen. Bitte wählen Sie diese aus.

Nun wählen Sie unter *„Sketch -> Bibliothek einbinden -> ZIP Bibliothek hinzufügen“* die „ESP32_BLE_Arduino-master“ aus (zu finden im GitHub Zip Ordner) und fügen diese in Arduino als Zip Bibliothek ein.



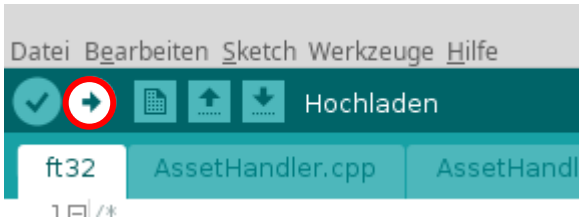
Verbinden Sie anschließend den eMalrob-FG über die serielle Schnittstelle (USB) mit Ihrem PC.



Wählen Sie nun unter *„Werkzeuge -> PORT“* den *„COM(3)“* aus.

Hinweis: Die Zahl nach „COM“ kann je nach eMalRob-FG und PC variieren.

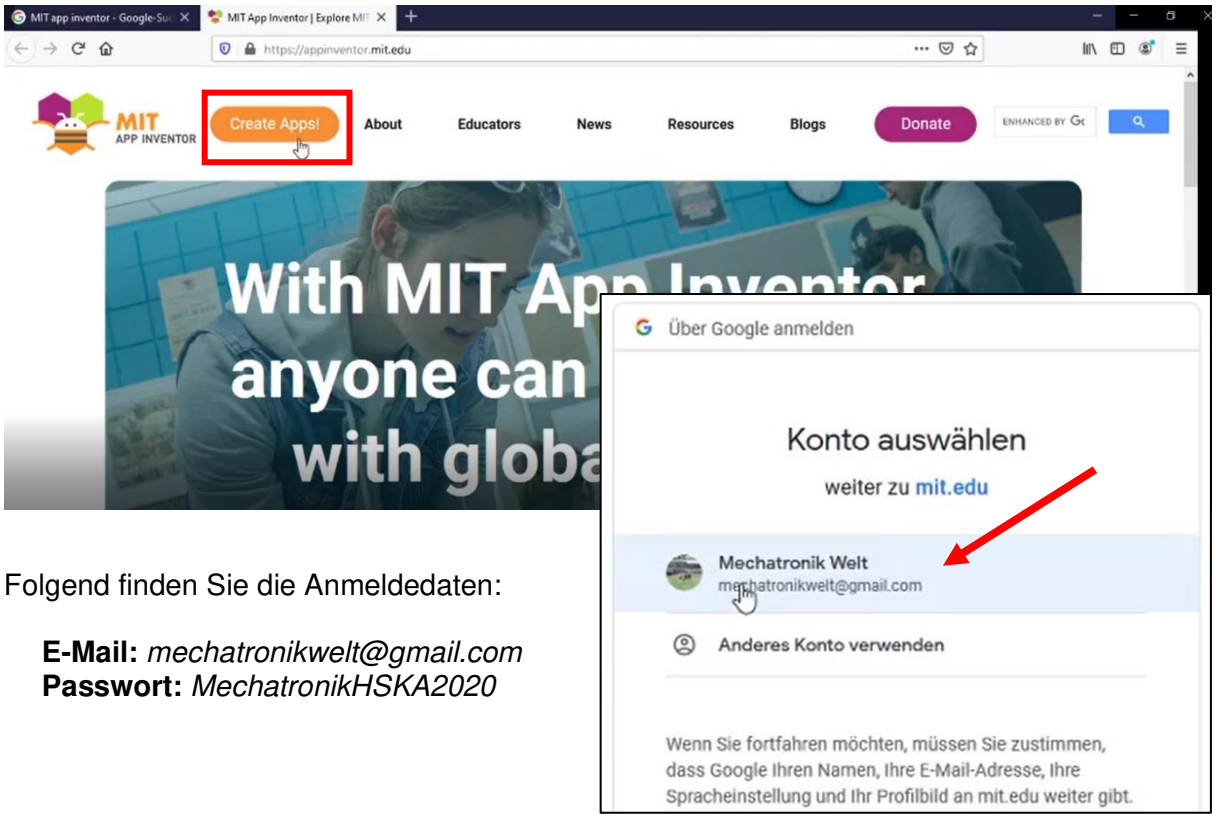
Nun laden Sie durch den *„Hochladen“* Button das Programm auf den ESP32 hoch.



Bitte nehmen Sie nun Ihr Smartphone und laden Sie in Ihrem Google Android Play Store die *„MIT AI2 Companion“* App kostenlos herunter.

Anschließend gehen Sie mit Ihrem Internetbrowser auf Ihrem PC auf die „App Inventor“ Webseite. Öffnen Sie diese unter folgendem Link:
https://appinventor.mit.edu/
(Dieser Link ist ebenfalls auf der im GitHub-Seite vorhanden.)

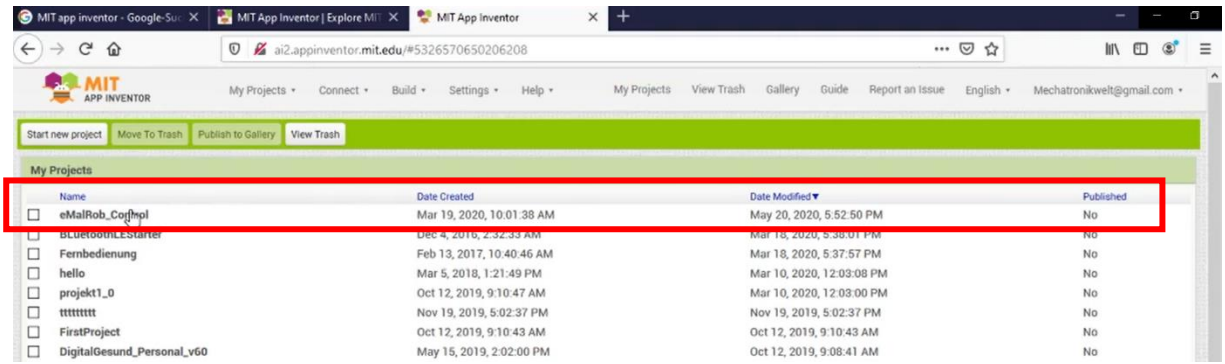
Unter *„Create Apps“* müssen Sie sich nun mit dem „Mechatronikwelt“ – Google-Account anmelden.



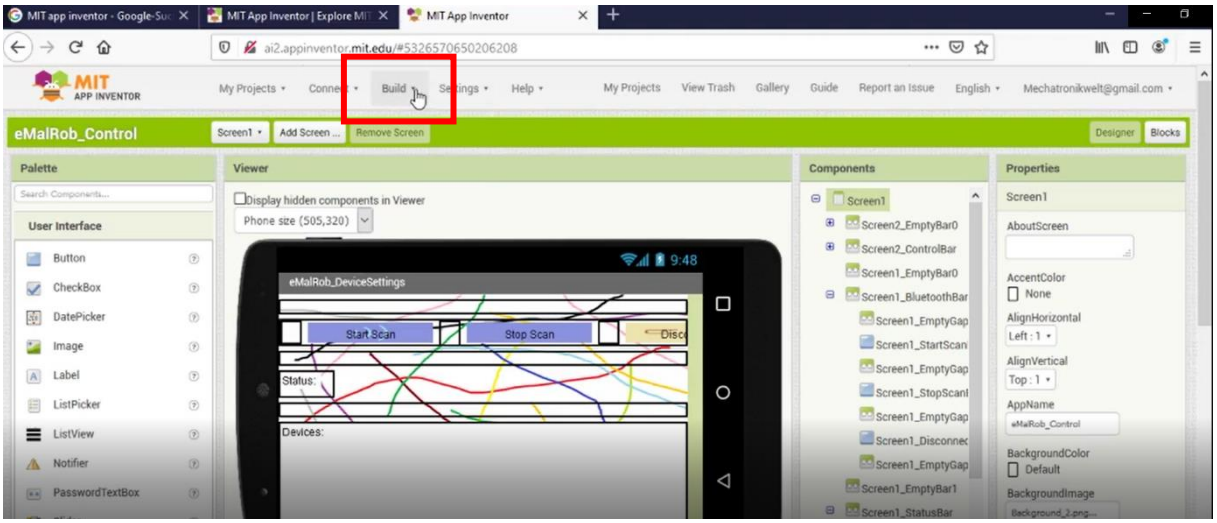
Folgend finden Sie die Anmeldedaten:

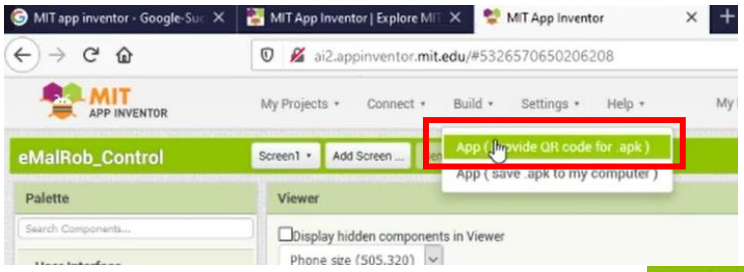
E-Mail: *mechatronikwelt@gmail.com*
Passwort: *MechatronikHSKA2020*

Klicken Sie nach dem Einloggen auf *„eMalRob_Control“*. Es öffnet sich jetzt die App-Oberfläche.



Zum weiteren Kompilieren der App klicken Sie unter dem Reiter *„Build“* auf *„App(provide QR Code for .apk)“*



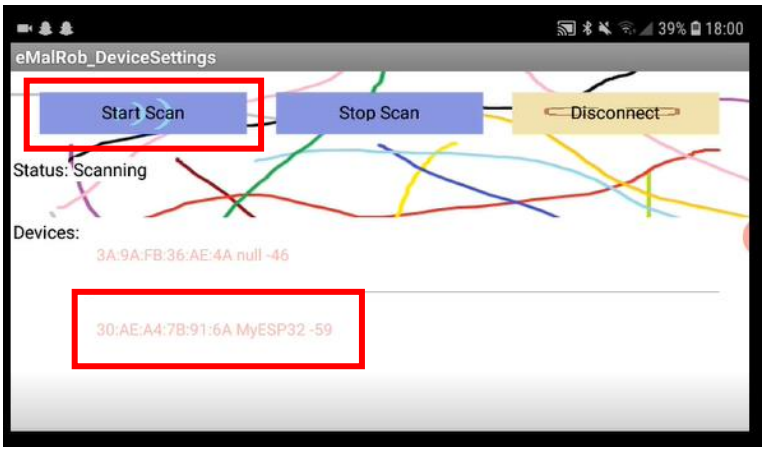


Öffnen Sie nun die zuvor heruntergeladene App „MIT AI2 Companion“ und scannen Sie den angezeigten QR Code ein.

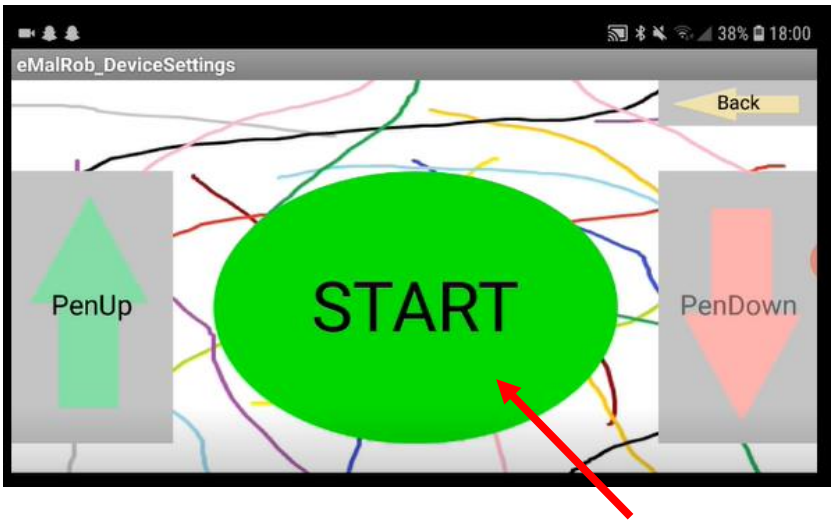
Hinweis: Der QR Code ist lediglich für 2 Stunden gültig.

Nun wird die .apk Datei geöffnet. Bitte klicken Sie anschließend „installieren“. Ist die Installation der App vollzogen, fahren Sie bitte mit „öffnen“ fort.

Schlussendlich müssen Sie noch die Bluetooth Verbindung von Ihrem Smartphone zum eMalRob-FG einrichten. Hierzu klicken Sie auf „Start Scan“ und wählen schließlich den ESP32, der aufgelistet wird, aus.



Wenn Sie nun den grünen „START“ Button auf Ihrem Smartphone betätigen, können Sie den eMalRob-FG anhand Ihres Smartphones durch Wenden und Kippen steuern. Mit „PenUp“ und „PenDown“ können Sie das Zeichnen auf der Oberfläche steuern.



S4 – Freie Programmierung extern auf dem Rechner

Das letzte und auch aufwendigste Szenario ist die Freie Programmierung, da diese ohne jegliche Hilfsunterstützung an Programmteilcodes fungiert. Diese Methode kann benutzt werden, um schon vorhandene Programmierkenntnisse weiter zu vertiefen. Die komplette Ansteuerung der Motoren und Funktionen kann größtenteils durch eine selbst entwickelte Art und Weise geschehen. Dies bedeutet, dass Sie den gesamten Code mit allen dazugehörigen Befehlen, wie zum Beispiel Bibliotheken, weiteren Quellcode- & Header-Dateien, PWM-Ansteuerungen der Motoren etc. selbst schreiben müssen.

Hilfe geben die verschiedenen Entwicklungsumgebungen der Studenten für den eMalRob-FG (EVAESP32, Cody++, usw.). Hierbei kann der freien Programmierung durch die Bausteinprogrammierung der bereits erwähnten Programmieroberflächen leicht unter die Arme gegriffen werden.

Diese Variante wird definitiv nicht als Einsteigervariante empfohlen.

Weitere Ideen/offene Aufgaben

In dieser Projektarbeit können Sie ganz detailliert die Kompilierungs- und Instandsetzungsszenarien des eMalRob-FG nachlesen und anhand des mitgelieferten GitHub Links ebenfalls ohne Probleme selbständig nachbilden und ausprobieren.

Für die Zukunft ist jedoch noch zu sagen, dass man sich darauf konzentrieren sollte, wie man den aus Bausteinzusammensetzung erstellte Cody++ Programmcode, welcher ebenso zeitgleich in die Programmiersprache C++ umgewandelt wird, fachgerecht und ohne große Mühen in den bereits von uns mitgelieferten OTA Code einbindet. Hierbei sollte man den OTA Code um alle notwendigen Implementierungen ergänzen, sodass letztendlich nur der jeweils neu programmierte „Setup“ Bereich und der „Loop“ Bereich kopiert und somit eingebunden werden muss. Eine Protokollierung der einzelnen Schritte, um zukünftig die Anwender noch leichter von der ersten Programmzeile zum letztendlichen Malen des Roboters zu führen, ist hierbei gewünscht

Der zweite Schritt wäre eine strukturierte Analyse der WLAN-Router. Hierbei ergab sich das Problem, dass die OTA Übertragung an manchen Routern problemlos funktionierte, an anderen Routern jedoch Fehler auftauchten. Hierbei ist die detaillierte Auflistung der richtigen Routereinstellungen zu überprüfen und diese fachgerecht zu dokumentieren.

Fazit

Abschließend können wir aus unserer Perspektive sagen, dass uns das Projekt sowohl Einblicke in das Programmieren mit Arduino, als auch das fachgerechte Verwenden eines ESP32 gelehrt hat.
Das mechatronische Gesamtprojekt „eMalRob-FG“ ist definitiv ein erfolgreiches Projekt für uns, um unser technisches Know-How in Sachen Programmierung zu stärken und ebenfalls nicht zu vernachlässigen, ergab sich in der kurzen Projektzeit eine hervorragende Zusammenarbeit im gesamten eMalRob-FG Team. Hierbei konnten wir anhand einer aktiven Kommunikation untereinander, sowohl unser soziales Teammanagement, die Teamarbeit an sich und natürlich die gemeinsame Planung enorm verbessern.

Schlussendlich nimmt der eMalRob-FG immer mehr Form an, ein sehr praktisches und zudem kostengünstiges Tool zum Lerneinstieg in die Programmierung, zu werden.

Unserer Meinung nach, ist es toll einen erheblichen Beitrag hierzu leisten zu dürfen und hoffen natürlich, dass dieses „Von Studenten für Schüler“ Projekt vielen Schülern, den in Zukunft immer wichtiger werdenden Aspekt der Programmierung, erleichtern wird und sie sich somit spielerisch zu einem Programmierliebhaber entwickeln können.