JOHANNES KEPLER
UNIVERSITÄT LINZ

Erstellt von

Tomislav Dodig

Emin Mehic

Hannah Fehringer

Angefertigt am

Institut für Wirtschaftsinformatik – Communications Engineering

Betreuer / Betreuerin

o.Univ.-Prof. Dipl.-Ing. Dr. Christian Stary

Claudia Kaar, BSc MSc

April 2020

KRISENMANAGEMENT MIT DEM INTERNET-OF-THINGS BEWÄLTIGEN

Eine M5Stack- und Blockly-Anwendung

Diese Anleitung zeigt den Einsatz von Internet-of-Things-Technologien zur Kontrolle von Zugängen. Ziel dabei ist die anschauliche, begreifbare Wissensvermittlung im Rahmen handlungsgeleiteter Lernprozesse. Die Ausführungen richten sich an Verantwortliche in Aus- und Weiterbildungseinrichtungen zur Integration von IoT-Szenarien in ihre Bildungsarbeit.

JOHANNES KEPLER
UNIVERSITÄT LINZ
Altenberger Straße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

INHALTSVERZEICHNIS

1. Einführung 3
2. Tangibles Internet-of-Things Starter Kit und Komponenten 4
3. Vorbereitung der Entwicklungsumgebung 9
4. Internet-of-Things-System – Einrichtung und Betrieb 10
   4.1. M5Burner 10
   4.2. UIFlow 12
   4.3. MIT App Inventor 16
      4.3.1. Control 19
      4.3.2. Variablen 20
      4.3.3. Layout 21
      4.3.4. Connectivity 21
      4.3.5. Texting 22
      4.3.6. Media 24
   4.4. Verwendung der erstellten Applikation über ein Endgerät 25
5. Realisierung des Anwendungsfalls 26
   5.1. Systemarchitektur (M5 Stack) 27
   5.2. Programm - Python Code (Version 1) 37
   5.3. Programm - Python Code (Version 2) 43
   5.4. Anwendungsfall-Logik (App Inventor) 49
   5.5. Programm – App Inventor Code 58
   5.6. Mobile Applikation – Überblick 61
6. Abbildungsverzeichnis 62

1. Einführung

Die Entwicklungen rund um Internet-Technologien halten in allen Bereichen der Gesellschaft Einzug. Dieser Anwendungsfall setzt sich mit dem sogenannten Internet-of-Things (IoT) und einigen damit verbundenen Methoden und Werkzeugen auseinander sowie der Erstellung einer Mobilen Applikation. Die in der Folge vorgestellten und genutzten Komponenten und Software können eingesetzt werden, um in Krisensituationen mittels mobiler Applikation Benachrichtigungen zu erhalten und dementsprechend reagieren zu können. Die in der Anleitung gezeigten Schritte erlauben die Konstruktion eines Krisenmanagmentsystems, das beispielsweise den Benutzer informiert falls im eigenen Haus Bewegungen stattfinden oder die Luftfeuchtigkeit sich dramatisch erhöht. Zusätzlich wird es zwei Versionen von diesem Krisenmanagementsystem geben. In der ersten Version werden die Units Laser RX und Laser TX eingesetzt, wohingegen in der Version 2 auf die Light Unit gesetzt wird. Das Krisenmanagementsystem besitzt dabei die Fähigkeiten,

- Bewegung im Raum zu erkennen
- Luftfeuchtigkeit zu messen
- Temperatur zu messen
- Luftdruck zu messen
- mittels Lichtschranke zu erkennen, ob eine Gewisse Grenze durchbrochen wurde oder nicht (Version 1)
- Licht im Raum zu erkennen (Version 2)

Zur Umsetzung dieses Anwendungsfalles wird das IoT-System **M5Stack** verwendet. Es besteht aus angreifbaren Elementen, und zwar Sensoren, Kabelverbindungen und dem Steuerelement M5 Stack, welcher mit kleinem Bildschirm und einfachen Druckknöpfen ausgestattet ist. Zur Umsetzung praxistauglicher Anwendungen ist zusätzlich eine Programmierumgebung erforderlich. In unserem Fall wird die Programmierumgebung UIFlow mit der graphischen Programmiersprache Blockly eingesetzt. Der Vorteil dieser Entwicklungsumgebung ist, dass zum Arbeiten mit dem M5Stack und Blockly von UIFlow kein Vorwissen im Bereich digitaler Systeme notwendig ist. Zusätzlich wird eine reine Code Programmierung durch die Programmiersprache Python unterstützt.

Der M5Stack ist ein Microcontroller-Entwicklungsboard, ausgestattet mit einem Gehäuse, Grafikdisplay, microSD Steckplatz, USB-C und Lautsprechern sowie Erweiterungssteckern. An einen M5Stack können mehrere zusätzliche Komponenten mit unterschiedlichen Fähigkeiten, wie z.B. dem Erkennen von Bewegung, angeschlossen werden.

- Diese Komponenten werden als ***Units*** bezeichnet.
- Die Anschlüsse am M5Stack für diese Units werden als ***Ports*** bezeichnet.

Mit Hilfe dieser Elemente kann ein einfaches Programm entwickelt werden. Der M5Stack ist darauf ausgelegt, Units zu verbinden. Durch stufenweise Erweiterung lassen sich an- und begreifbar, und damit in überschaubaren Schritten, komplexe Systeme entwickeln.

Da der M5Stack auch mit Lego-Bausteinen kompatibel ist, bietet das System auch für Kinder und Jugendliche die Möglichkeit, Hemmschwellen zur Beherrschbarkeit von digitalen Systemen, und somit auch zur Programmierung, zu überwinden.

UIFlow ist gezielt darauf ausgelegt, den M5Stack zu programmieren, und zwar mittels der visuellen Programmiersprache ***Blockly*** oder textbasiert in Python. Blockly ist ein visuelles Drag-and-Drop Block-Stecksystem, mit dem die Funktionalität des M5Stack bearbeitet werden kann. Mit Blockly ist für das Programmieren wenig Schreibaufwand verbunden, sodass die Entwicklung durch einfaches Verschieben und sachorientiert erfolgen kann.

Für die Entwicklung der mobilen Applikation wird die MIT App Inventor Plattform verwendt, die ebenfalls wie UIFlow ein Drag-and-Drop Block-Stecksystem zur Programmierung anbietet.

Diese Anleitung ermöglicht einen einfachen Einstieg in die Verwendung des M5Stack und seiner Komponenten, sowie der Programmierumgebung UIFlow und der Erstellung einer mobilen Applikation durch die MIT App Inventor Platform. Sie hilft, einen Anwendungsfall mit den erwähnten Bestandteilen (M5Stack, UIFlow und MIT App Inventor) zu erstellen. Die Anleitung umfasst folgende Kapitel neben dieser Einführung in das Thema:

- Kapitel 2 stellt das Starter-Kit und seine Komponenten vor, sowie jene zusätzlichen Units, die für die Realisierung des Anwendungsfalls benötigt werden.
- Kapitel 3 fasst die Vorbereitung der gesamten Entwicklungsumgebung in Form einer Checkliste zusammen.
- Kapitel 4 zeigt die Vorbereitung des M5Stack und seiner wichtigsten Komponenten zur Entwicklung des Anwendungsfalls.
- Kapitel 5 beschreibt die Erstellung des Anwendungsfalls. Sowohl textuell als auch grafisch wird Schritt-für-Schritt die Entwicklung eines funktionstüchtigen IoT-basierten Krisenmanagementsystem gezeigt.

Das Krisenmanagementsystem wird somit nicht nur benutzerfreundlich entwickelt, sondern weist alle Eigenschaften auf, wie IoT-Systeme praktisch genutzt und im individuellen Umfeld eingesetzt werden können.

2. Tangibles Internet-of-Things Starter Kit und Komponenten

In diesem Abschnitt werden das zur Realisierung des Anwendungsfalls erforderliche M5Go IoT Starter Kit ESP32 sowie zusätzliche Komponenten erläutert und beschrieben.

Das Starter Kit von M5 enthält 6 verschiedene Units, einen M5Stack sowie Lego Bausteine. Dieses Starter Kit erleichtert, mit dem Konzept des Internet-of-Things (IoT) vertraut zu werden und wesentliche Komponenten kennenzulernen. Jede Komponente kann separat angeschafft werden, sodass der Einsatz von Komponenten in mehreren Etappen erfolgen kann.

Im Starter Kit enthalten ist:

M5Stack:

Diese Komponente ist die zentrale physische Komponente jedes IoT-Systems. An den M5Stack lassen sich alle Units anschließen. Dies ermöglicht die Kombination von mehreren Units.

Motion Sensor:

Diese Unit kann Bewegung erkennen.

ANGLE Sensor:

Diese Unit erlaubt die Erkennung ausgewählter Werte.

Grove Hub:

Diese Unit erlaubt es, bis zu drei Units an einen Port anzuschließen.

IR Remote:

Diese Unit ermöglicht die Kommunikation mit Geräten über Infrarot-Strahlen.

RGB LED:

Diese Unit besitzt drei Lampen, die jeweils sämtliche Farben der RGB Skala annehmen können.

Environment Sensor:

Diese Unit kann die Temperatur, die Luftfeuchtigkeit und den Luftdruck messen.

Abbildung 1 gibt einen Überblick über die beschriebenen Komponenten (Units) des Starter Kit.

Abbildung 1: M5Go IoT Starter Kit ESP32

Für das Krisenmanagementsystem werden jedoch noch weitere Units benötigt:

Grove Hub:

Mit diesem Hub lassen sich bis zu drei Units an einen Port anschließen.

Abbildung 2: Grove Hub Unit Abbildung

pb.Hub:

An diesen Hub lassen sich bis zu sechs Units anschließen, jedoch mit der Einschränkung, dass diese nur für den Port B gedacht sind. Eine Besonderheit von diesem Hub ist es, dass er an Port A angesteckt werden muss.

Abbildung 3: Pb.Hub Unit Abbildung

Motion Sensor:

Diese Unit kann Bewegung erkennen.

Abbildung 4: Motion Sensor Unit Abbildung

Environment Sensor:

Dieser Sensor kann die Temperatur, die Luftfeuchtigkeit und den Luftdruck messen.

Abbildung 5: Environment Sensor Unit

Laser RX:

Diese Unit funktioniert nur in Kombination mit der Unit Laser TX. Laser RX empfängt den Infrarotstrahl von Laser TX.

Abbildung 6: Laser RX Unit Abbildung

Laser TX:

Diese Unit funktioniert nur in Kombination mit der Unit Laser RX. Diese Unit sendet einen Infrarotstrahl, der von der Laser RX Unit empfangen werden kann.

Abbildung 7: Laser TX Unit Abbildung

Sobald alle Komponenten für das Krisenmanagementsystem richtig zusammengesteckt wurden, ergibt sich mit dem M5Stack und den notwendigen Units der in Abbildung 9 gezeigte Aufbau. Das Zusammenstecken der Units ist im Abschnitt Bauanleitung erklärt. Die mobile Applikation wird am Ende, wie in Abbildung 8 dargestellt, aussehen.

Abbildung 9: M5Stack und die dazugehörigen Units für das Krisenmanagmentsystem

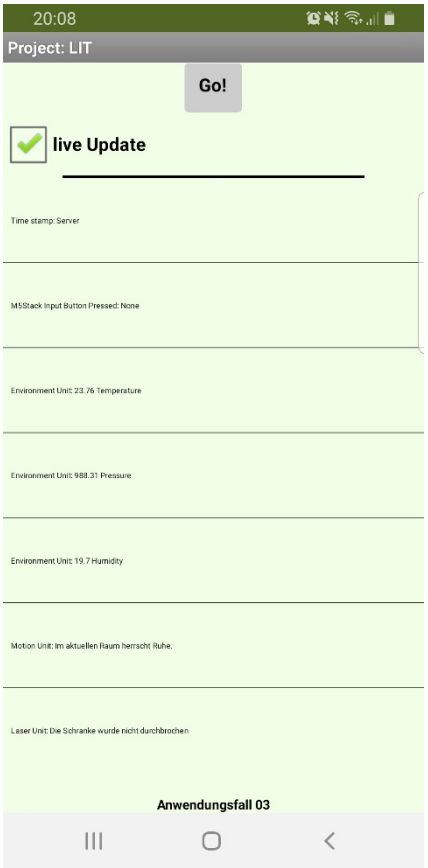


Abbildung 8: Optische Darstellung der mobilen Applikation

3. Vorbereitung der Entwicklungsumgebung

Die folgende Checkliste erleichtert den Einstieg in die IoT-Anwendungsentwicklung. Die Liste enthält sämtliche notwendigen Vorkehrungen, um das Krisenmanagementsystem mit dem M5Stack zu realisieren.

- ☐ Installation von M5Burner auf Computersystem - M5Burner ist unter https://m5stack.com/pages/download zu finden
- ☐ Verwendung des M5Burner um auf dem M5Stack die aktuellste Firmware zu installieren (mehr Info unter M5Burner)
- ☐ Installieren von Treiber CP2104 Driver des M5Stack – dieser ist unter https://m5stack.com/pages/download zu finden
- ☐ Installation von lokaler UIFlow-Programmierumgebung auf Computersystem (Mehr Info unter UIFlow)
- ☐ Mobile Applikation mittels MIT App Inventor erstellen (Mehr Info unter http://appinventor.mit.edu/)
- ☐ Verwendung der Anleitung Krisenmanagement mit der Internet-of-things Anleitung, um mit den Units, UIFlow, MIT App Inventor Plattform und M5Stack vertraut zu werden.

4. Internet-of-Things-System – Einrichtung und Betrieb

Der M5Stack bildet gemeinsam mit den in diesem Abschnitt vorgestellten Software-Entwicklungskomponenten das Internet-of-Things-System. Zur Entwicklung und zum Betrieb wird der M5Stack mit dem Computersystem über ein USB-C Kabel angeschlossen. Sobald der M5Stack eingeschaltet wird, ist es notwendig, den richtigen Modus auszuwählen, um eine Verbindung mit dem Computersystem herzustellen. Dafür drückt man Knopf C, um in die Setup Kategorie zu gelangen. Von dort wählt man „Switch to USB mode“ um das Gerät richtig einzustellen. In Abbildung 12 wird der M5stack mit seinen verschiedenen Ports und Knöpfen gezeigt.

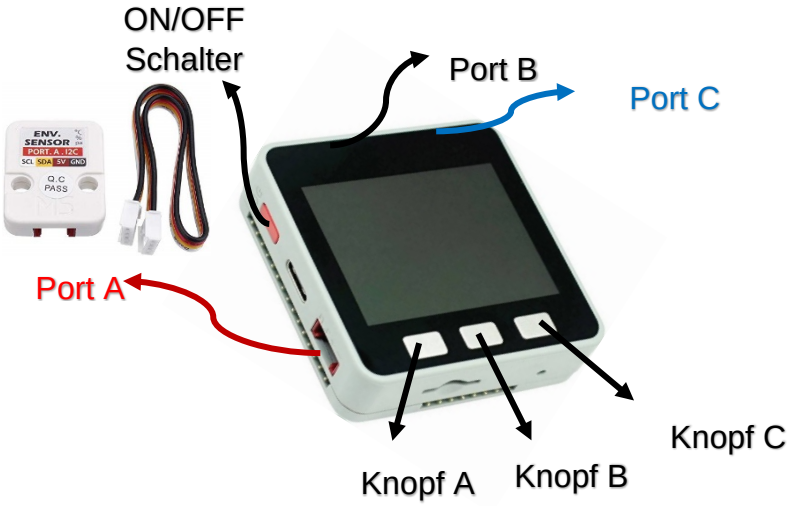


Abbildung 10: Beschriftung vom M5Stack

4.1. M5Burner

Bei der allerersten Verwendung des M5Stack muss man den M5Burner verwenden. Den M5Burner kann man von der Seite https://m5stack.com/pages/download laden. Durch den M5Burner kann man die Firmware auf dem M5Core installieren. Dieser muss dieselbe Version haben wie UIFlow (wenn man damit arbeitet).

Den M5Burner kann man nach dem Download über die Datei „M5Burner.exe“ ausführen, wie in der nächsten Abbildung gezeigt.

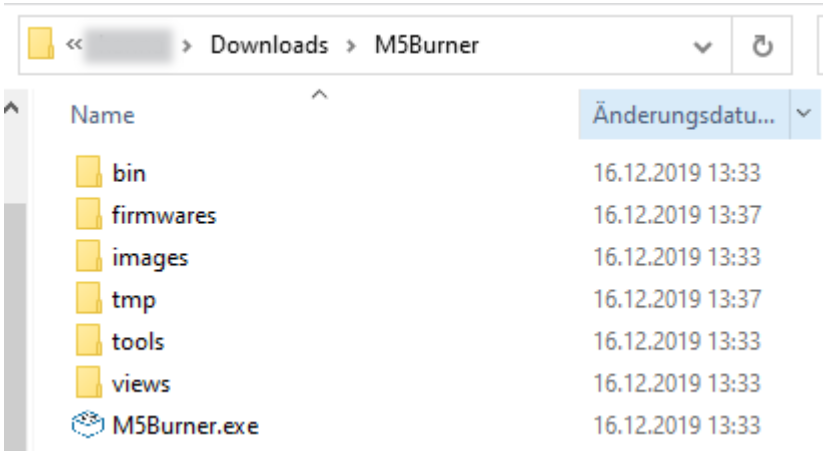


Abbildung 11: Ordnerstruktur vom M5Burner

Sobald man den M5Burner geöffnet hat, muss man die richtige UIFlow-Version laden. Dafür muss man auf das Download-Symbol neben der ausgewählten UIFlow-Version klicken. Danach sind der M5Stack über das USB-C Kabel an den Laptop/PC anzuschließen und der richtige COM-Anschluss auszuwählen. Empfehlenswert ist es hier die Version „UIFlow-v1.4.5“ auszuwählen, weil diese die Möglichkeit erlaubt innerhalb der grünen Felder den WLAN-Name sowie das dazugehörige Passwort einzugeben. Für den Anwendungsfall ist es besonders wichtig, dass der M5Stack als auch das dazugehörige Smartphone (oder Tablet) im gleichen Netzwerk ist.

Anschließend kann man auf Burn klicken und die Firmware wird auf dem M5Stack installiert. Die Darstellung vom M5Burner wird in Abbildung 5 gezeigt.

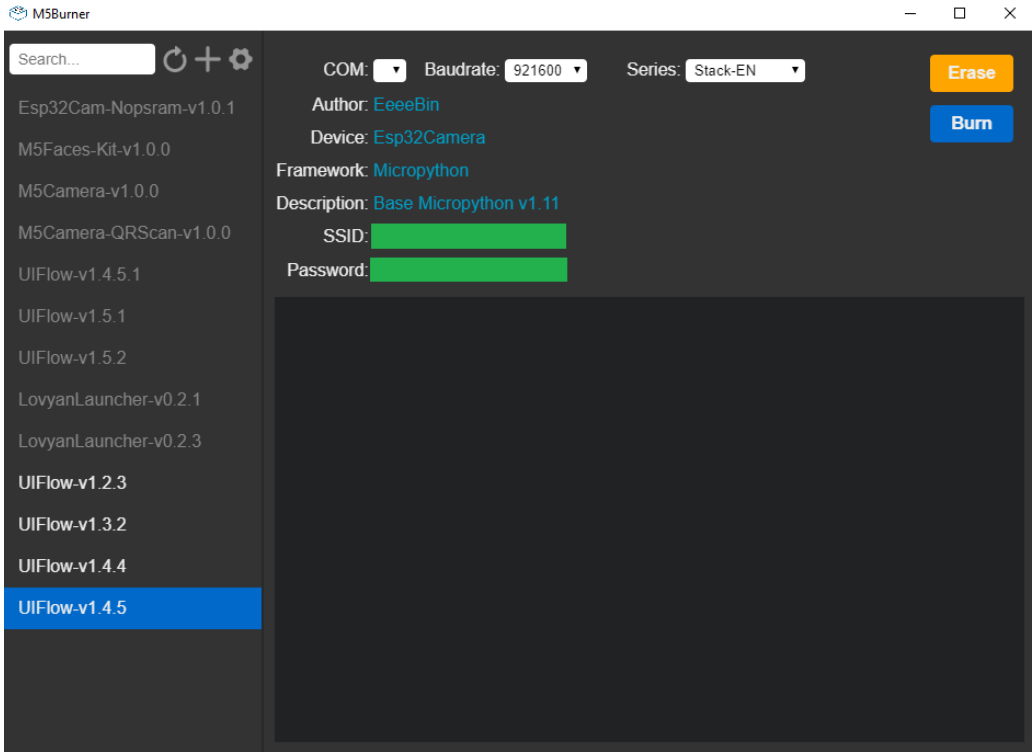


Abbildung 12: Benutzeroberfläche vom M5Burner

Es kann vorkommen, dass das Computersystem den M5Stack nicht erkennt. Sollte dies geschehen, kann der Treiber „CP2104 Driver“ für den M5Stack, ebenfalls von der Seite https://m5stack.com/pages/download geladen werden.

4.2. UIFlow

Die lokale Programmierumgebung von UIFlow kann von https://m5stack.com/pages/download geladen werden. Innerhalb der UIFlow-Oberfläche wird beim Starten der lokalen Programmierumgebung nach einem verbundenen M5Stack gefragt, welcher in der COM: Schaltfläche ausgewählt wird, wie in Abbildung 15 zu sehen ist.

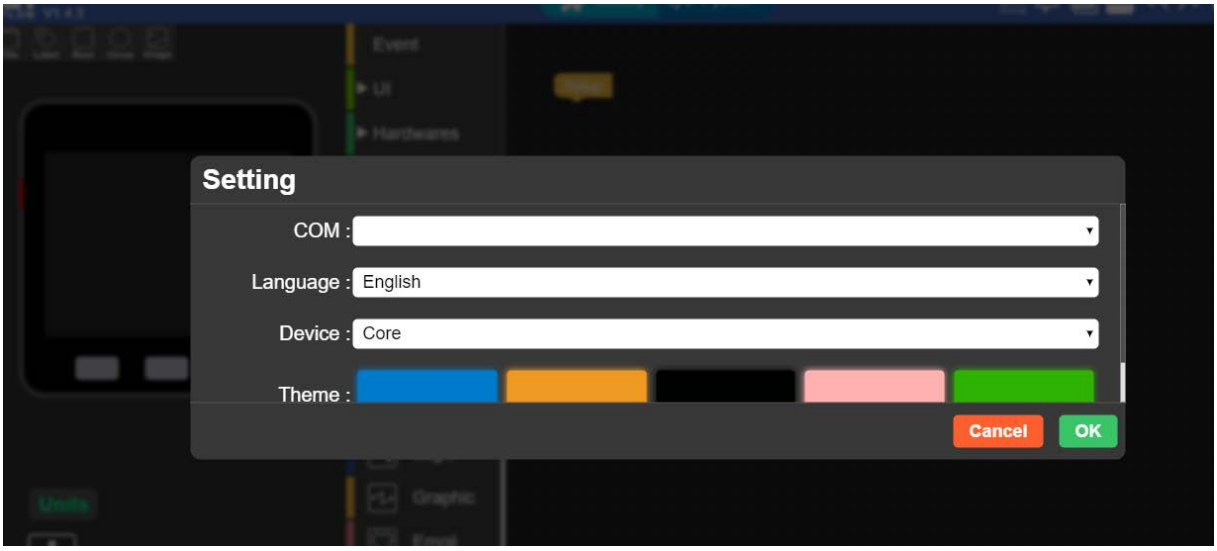


Abbildung 13: Setting Auswahl vom UIFlow

Üblicherweise wird der korrekte COM-Anschluss automatisch angezeigt. Sollte dies jedoch nicht der Fall sein, so kann man über den Gerätemanager des Computersystems herausfinden, welcher COM-Anschluss in UIFlow zu verwenden ist. Dafür klickt man mit der rechten Maustaste auf das Windows Symbol im unteren linken Bereich des Desktop und anschließend wählt man „Geräte-Manger“ aus, wie in der folgenden Abbildung zu sehen ist.

Unter „Anschlüsse (COM & LPT)“ kann man dann herausfinden, über welchen COM der M5Core angesprochen werden kann. Diese Information findet man, sobald „Silicon Labs CP210x USB to UART Bridge (…)“ unter den Anschlüssen aufscheint.

Abbildung 14: Die Ansicht, nachdem ein Rechtsklick auf das Windows Symbol ausgeführt wurde

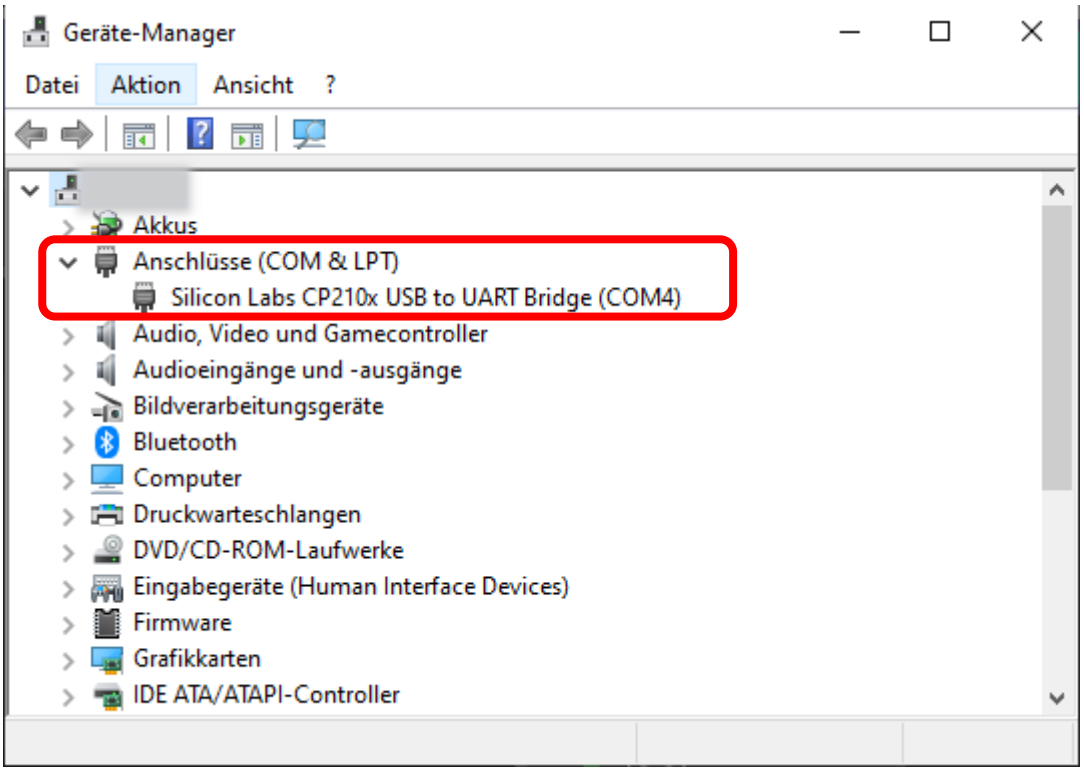


Abbildung 15: Anzeige der verschiedenen Anschlüsse am Computer

Dieser COM-Anschluss muss dann auf der Startseite von UI-Flow eingegeben werden.

Nach einer erfolgreichen Verbindung ist die Nutzung vom M5Stack und UIFlow vorbereitet. Dies erkennt man, wenn am unteren linken Rand des Bildschirms der Status „Connected“ sichtbar ist. Diese Anzeige ist in der nächsten Abbildung zu sehen.

Abbildung 16: M5Stack-Verbindung wird erfolgreich angezeigt.

Danach ist die Nutzung der UIFlow-Plattform möglich – die ersten Schritte für die Entwicklung einer eigenen Anwendung sind gesetzt. In Abbildung 19 werden die Benutzungsschnittstelle dargestellt und die wichtigsten Komponenten hervorgehoben.

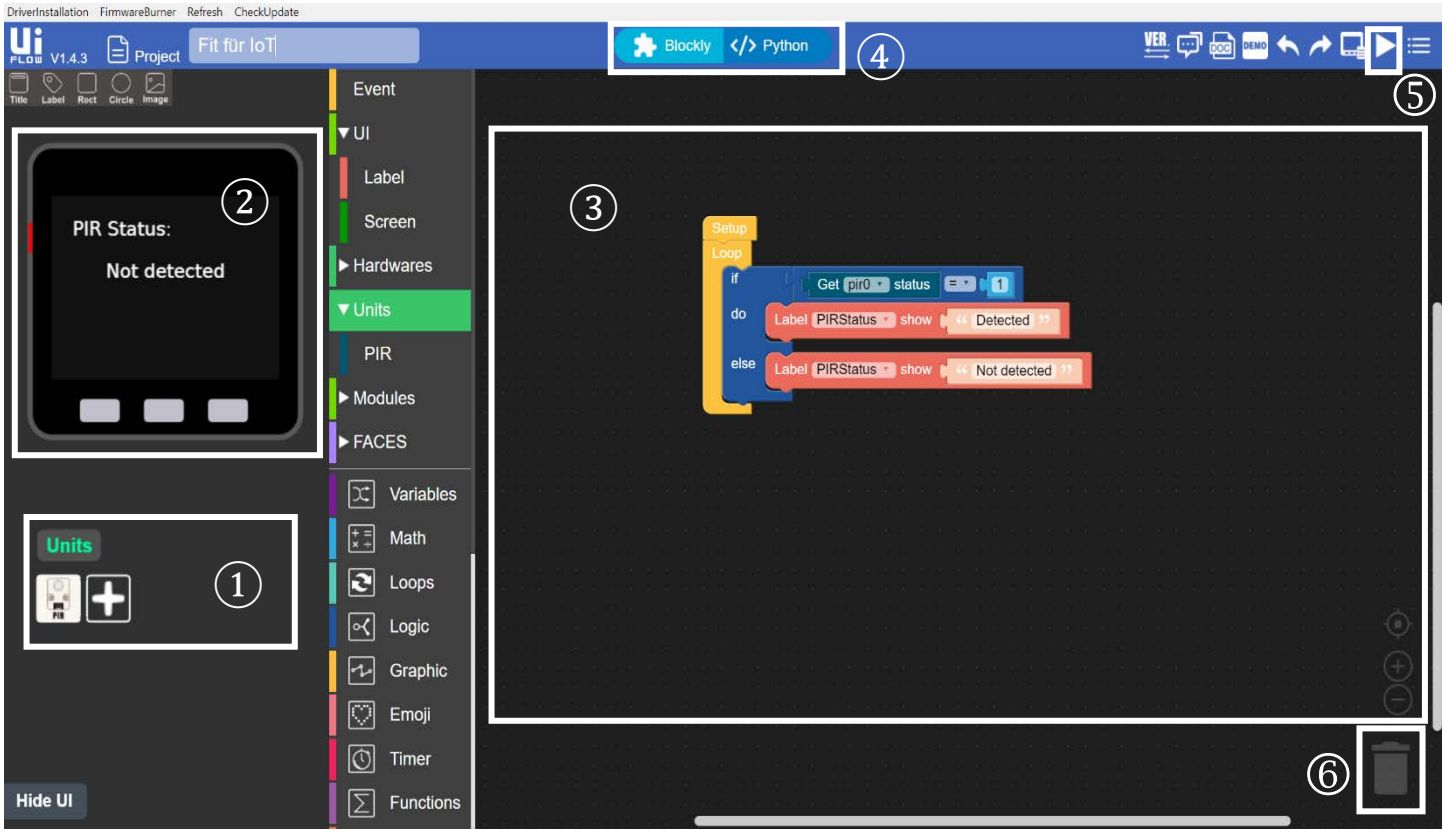


Abbildung 17: Erklärung zur UIFlow Oberfläche (anhand eines PIR Beispiel).

1. Hier können jene Units hinzugefügt werden, die man programmieren möchte. Nähere Informationen zu Units siehe Abschnitt Units.
2. Die Oberfläche des M5Core kann hier kreiert werden. Kreise, Texte, Quadrate und Bilder können per Drag-and-Drop eingefügt werden.
3. Dies ist der Programmierbereich (Coding Area). Hier erstellt man per Drag-and-Drop seinen Code, den man später ausführen möchte, zusammen.
    a. Blöcke sind Bausteine in verschiedensten Farben mit verschiedensten Funktionen, die benutzt werden, um eine Ablauflogik zu programmieren. Abhängig davon, welche Funktion man benötigt, kann man einen Block per Drag-and-Drop auf der Oberfläche einfügen. Im Beispiel werden Blöcke aus dem Event-, PIR- und Label-Block verwendet.
4. Hier kann man zwischen den Programmiersprachen Blockly und Python wechseln.
    a. Für diesen Anwendungsfall wird die Programmierung unter Python herangezogen.
5. Damit spielt man den Code temporär auf den M5Core, welcher diesen ausführt, sofern er korrekt zusammengestellt wurde.
6. Per Drag-and-Drop kann man zu diesem Papierkorb Blöcke hinziehen, die man nicht mehr benötigt, damit sie gelöscht werden.

4.3. MIT App Inventor

Die Webseite von MIT App Inventor ist unter folgendem Link zu finden: https://appinventor.mit.edu/.

Nach erfolgreichem Aufrufen der Webseite, klicke anschließend auf das Feld „Create Apps!“.

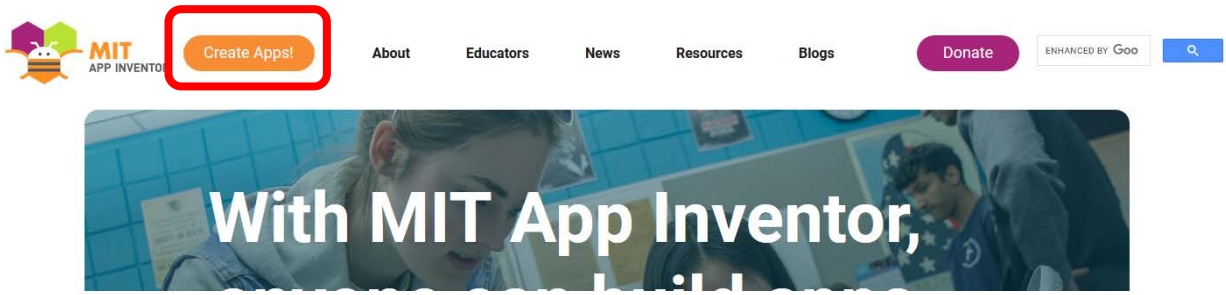


Abbildung 18: Die offizielle MIT App Inventor Webseite

Danach registriere dich auf der Seite oder verwende deinen Google Account, um automatisch registriert zu werden und akzeptiere die Benutzungsbedingungen. Anschließend erstelle ein neues Projekt unter „Start new project“ und verleihe der Applikation einen passenden Namen.

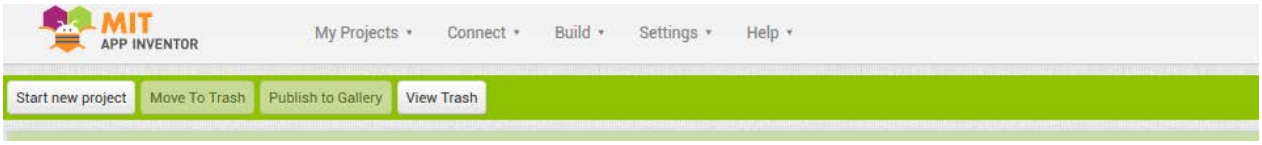
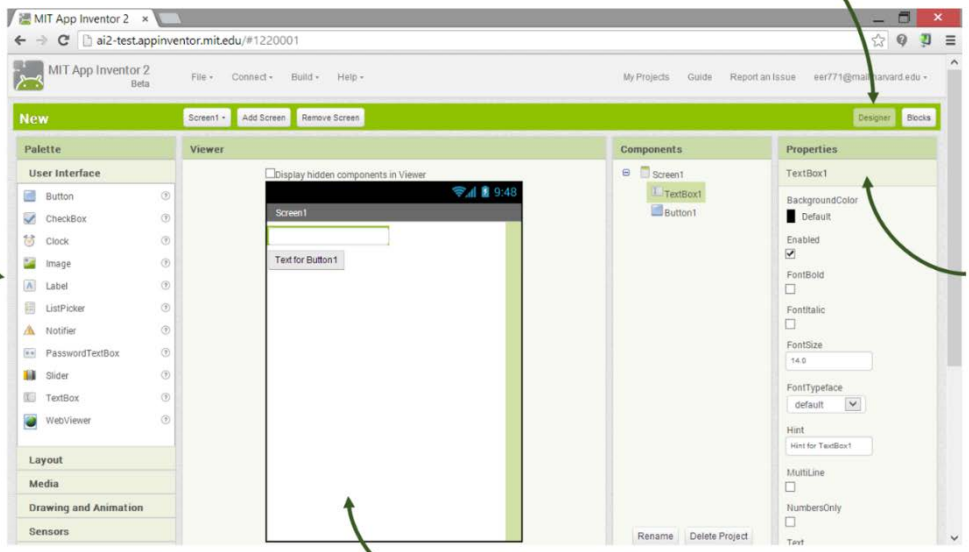


Abbildung 19: Die Darstellung, wenn ein neues App Projekt gestartet wird

Danach ist die Nutzung der MIT App Inventor Plattform möglich. In Abbildung 19 wird das Design vom App Inventor dargestellt. Nachfolgend in Abbildung 20 wird die Ansicht vom App Inventor vorgestellt und in Abbildung 21 wird der Block Editor erklärt. Diese Komponenten werden unterhalb des Bildes noch genauer erklärt.

Palette: Hier werden alle Komponenten angezeigt. Mittels Drag & Drop werden die Komponenten hinzugefügt.

Designer Knopf: Wechsle von der Block Ansicht zur Designer Ansicht.



Properties: Die Eigenschaften der Komponenten lassen sich hier verändern. Klicke auf ein Objekt welches sich bereits auf der Oberfläche befindet und verändere die Farbe, Größe etc.

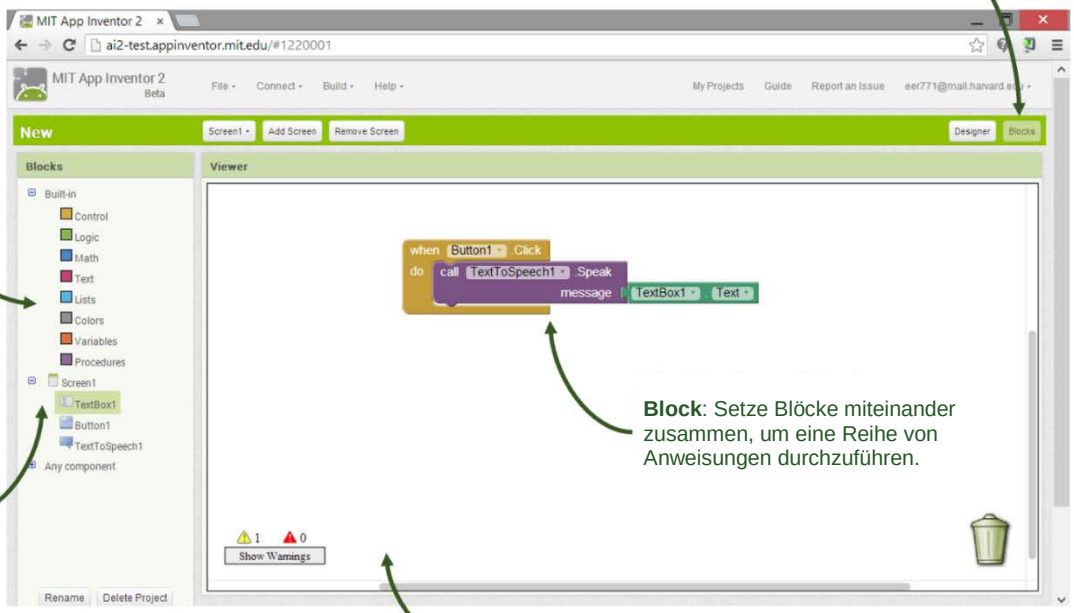
Viewer: Die optische Darstellung deiner Applikation. Ziehe eine Komponente aus der Palette herein, um zu sehen wie deine Applikation aussehen wird.

Abbildung 20: Die Designer Ansicht der MIT App Inventor Plattform

Auf der nächsten Abbildung wird der Block Editor näher vorgestellt.

Built-In Drawers: Blöcke für generelle Anweisungen werden hier angezeigt.

Block Knopf: Wechsle von der Design Ansicht zur Block Ansicht.



Block: Setze Blöcke miteinander zusammen, um eine Reihe von Anweisungen durchzuführen.

Specific Component: Blöcke für bestimmte Komponenten, die du bereits hereingezogen hast, werden hier angezeigt.

Viewer: Ziehe Blöcke vom Pannel links in den Viewer, um eine Verbindung mit den Anweisungen herzustellen.

Abbildung 21: Die Block Ansicht der MIT App Inventor Plattform

Die Blöcke, die man zum Programmieren verwendet, befinden sich in dem Panel auf der linken Hälfte der MIT App Inventor Oberfläche. Dort kann zuerst eine Metakategorie ausgewählt werden und anschließend der notwendige Block per Drag-and-Drop in den Viewer gezogen werden. Zur Veranschaulichung wird der Screen1-Block in Abbildung 22 verwendet.

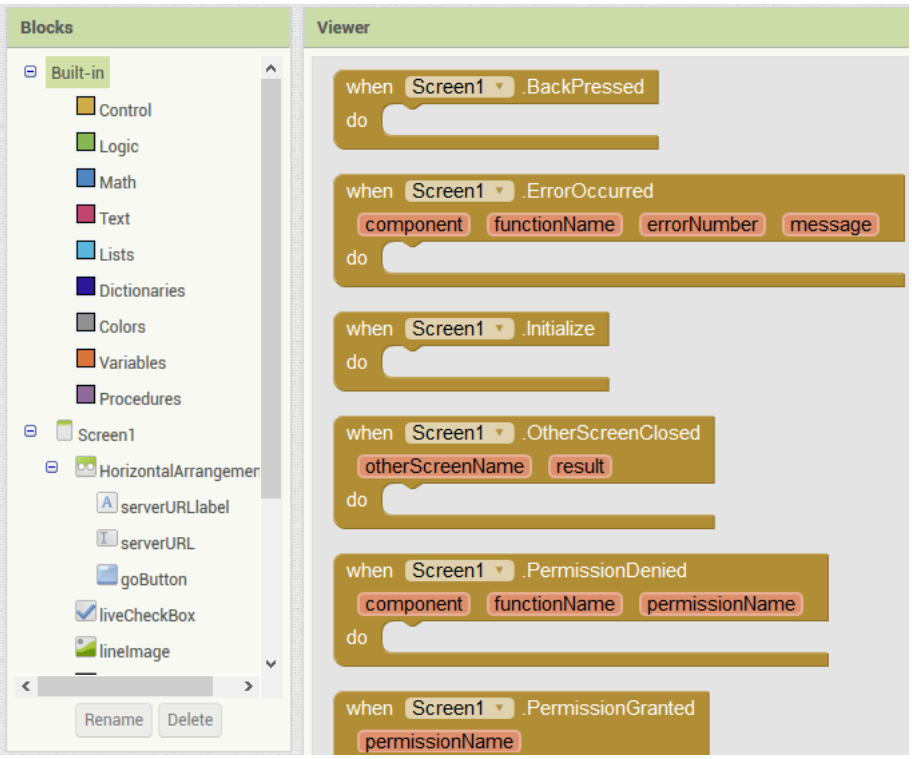


Abbildung 22: Anzeige der verschiedenen Blöcke in der Designer Ansicht

Um mit dem Programmieren sinnvoll starten zu können, sollte man jedoch davor noch gewisse Konzepte kennen. Diese Konzepte werden im weiteren Verlauf beschrieben.

4.3.1. Control

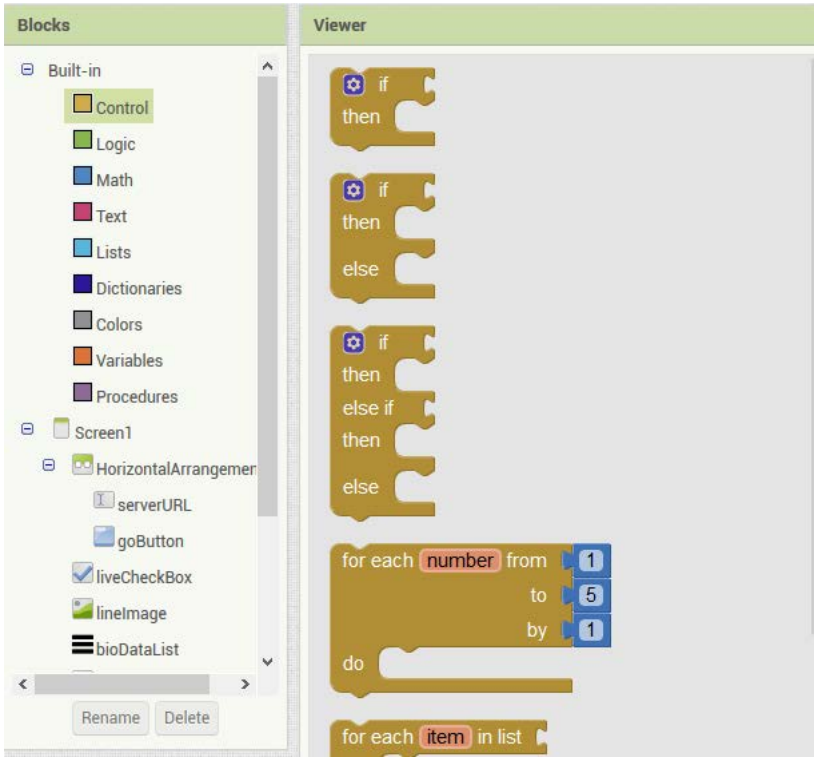


Abbildung 23: Control Block Ansicht

Eine App funktioniert nicht ohne Regeln und Anweisungen. Aus diesem Grund sind gerade if-Abfragen essenziell, um ein Programm zu schreiben. Unter der Control Kategorie hat man die Auswahl zwischen mehreren Iterationen von if-Abfragen.

4.3.2. Variablen

Variablen stellen sozusagen das Gedächtnis eines Programms dar. Sie können Werte speichern, die im Zuge der Ausführung eines Programms geändert bzw. verwendet werden können, und zwar durch den Aufruf des Namens der Variable.

Abbildung 24: Variablen Ansicht

Beispiele unterschiedlicher Variablen sind in der nächsten Abbildung zu sehen.

Abbildung 25: Ein Beispiel verschiedener Variablen

4.3.3. Layout

Unter Layout können verschiedene Layout-Systeme ausgewählt werden, um den verschiedensten Komponenten eine Struktur zu verleihen. Somit stellt man sicher, dass gewisse Textfelder, Knöpfe oder sonstige Komponenten geordnet dargestellt werden.

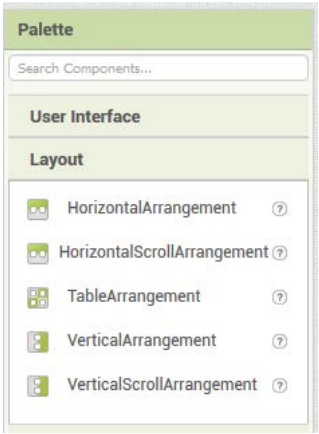


Abbildung 26: Layout Ansicht

4.3.4. Connectivity

In der Connectivity verleiht man der App Verbindungsmöglichkeiten. Für das Krisenmanagementsystem ist die Komponente „Web“ notwendig.

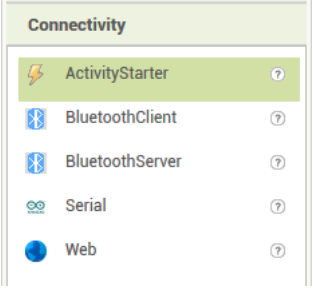


Abbildung 27: Connectivity Ansicht

4.3.5. Texting

Als Texting werden die Textplatzhalter auf der Applikation bezeichnet, mit denen man gewisse Werte oder Strings eingeben kann.

Um ein Texting auf der Applikation anzuzeigen, muss man dieses von der linken oberen Ecke unter „Social“ – „Texting“ per Drag-and-Drop auf die Viewer Fläche ziehen. Diese Auswahlmöglichkeiten findet man in der Designer-Ansicht.

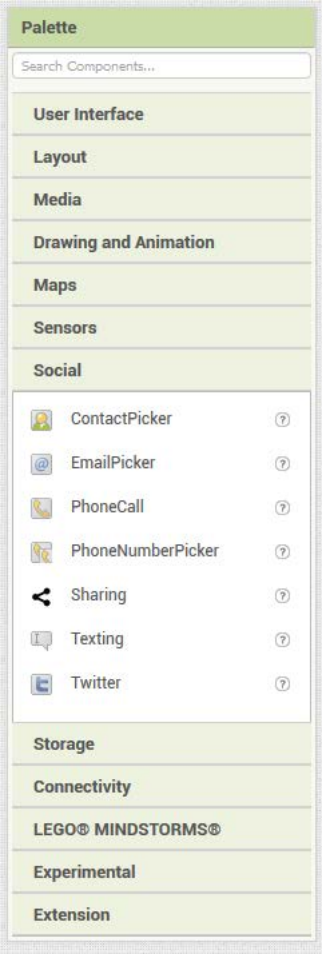


Abbildung 28: Social Ansicht, in der auch Texting vorhanden ist

Danach erscheint in der Blocks-Ansicht eine Auswahlleiste für die Blöcke unter Viewer eine Subkategorie. Somit kann man jetzt die Blöcke auswählen, mit denen man die Ausgabe auf der App verändern kann. Ein solcher Block wird in der Abbildung 29 gezeigt.

Abbildung 29: Ansicht vom erstellten HorizontalArrangement

Das Prinzip für andere Funktionen wie Label, Button, CheckBox, Image, ListView, Notifier und allen anderen Funktionen in der Palette ist das gleiche wie für Texting.

4.3.6. Media

Über Media können spezifische Bilder hochgeladen werden, die anschließend in der Applikation verwendet werden können. Um diese hochzuladen klickt man zuerst auf „Upload File…“ wie in der nächsten Abbildung dargestellt.

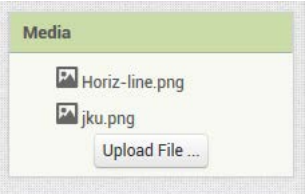


Abbildung 30: Media Ansicht

Danach wird ein neues Fenster aufgerufen in dem durch Klick auf den „Durchsuchen…“ Knopf, die gewünschte Datei ausgewählt werden kann.

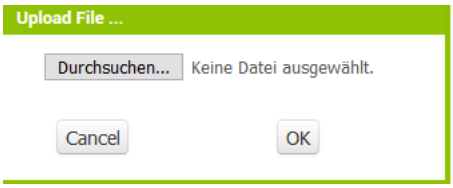


Abbildung 31: Fenster, welches geöffnet wird, um ein Bild hochzuladen

4.4. Verwendung der erstellten Applikation über ein Endgerät

Nachdem die Inhalte der MIT App Inventor Plattform vorgestellt wurden, ist der nächste Schritt die erstellte Applikation auf dem gewünschten Endgerät (Handy oder Tablet) auszuprobieren.

1. Lade und installiere die MIT AI2 Companion Applikation auf dein Endgerät herunter. Scan dafür diesen QR Code oder lade die App vom Play Store herunter. (Klicke hier für den Link)

   Abbildung 32: QR Code für die MIT AI2 Companion App

   Nach dem Herunterladen gehe die Installationsanweisung von der MIT AI2 Companion App durch. Die MIT AI2 Companion App muss nur einmal installiert werden. Nach erfolgreicher Ausführung kann die App auf dem Endgerät verwendet werden, wann immer ein neues Projekt gestartet wird.

2. Verbinde deinen Computer und dein Endgerät mit dem gleichen WiFi Netzwerk.

3. Öffne App Inventor Projekt und verbinde dich mit deinem Endgerät. Anschließend wähle „Connect“ und „AI Companion“ von der oberen Menü Fläche (siehe Abbildung 33).

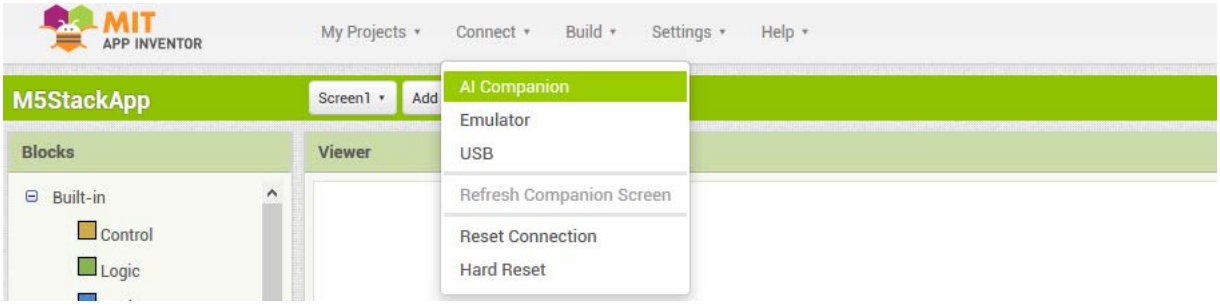


Abbildung 33: Auf AI Companion klicken, um die Applikation zu erstellen

4. Im nächsten Schritt wird ein QR Code auf deinen Computer angezeigt, die du mit deinem Endgerät lesen kannst. Öffne die MIT AI2 Companion App mit dem Endgerät und klicke anschließend auf „Scan QR Code“ und scanne den QR Code, der auf dem Computer angezeigt wird.

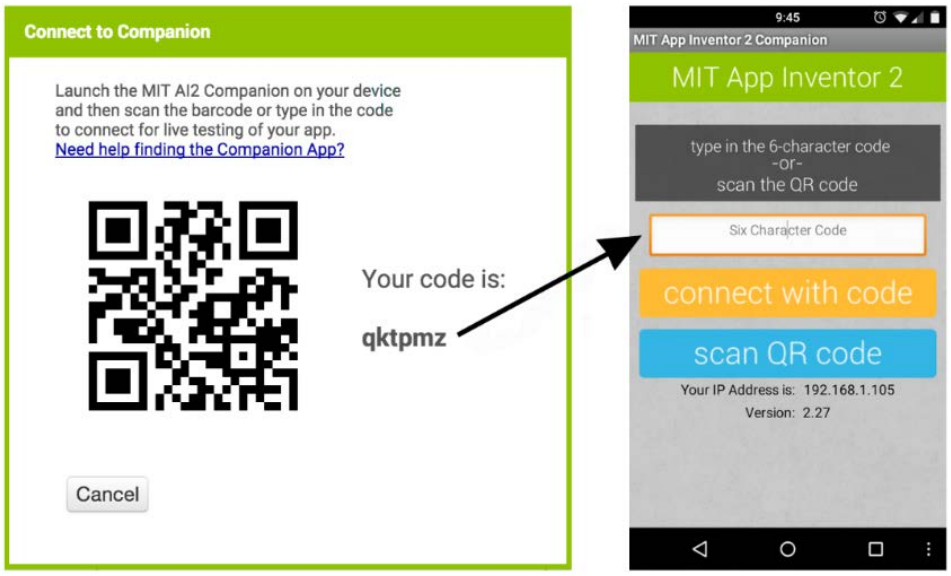


Abbildung 34: Das neue Fenster, dass den QR Code anzeigt, um diesen mit der MIT AI2 Companion App einzuscannen

Nach einigen Sekunden wird die erstellte App auf das Endgerät geladen und ist einsatzbereit.

5. Realisierung des Anwendungsfalls

Der Krisen-Anwendungsfall betrifft ein Inhouse-Krisenmanagementsystem welches zusätzlich über eine mobile Applikation kontrolliert werden kann. Er erlaubt, Informationen über Räume zu erhalten und informiert zu werden, wenn festgelegte Grenzen durchschritten werden. Egal ob junge Erwachsene oder besorgte Eltern, der Wunsch das eigene Haus in Sicherheit zu wissen und benachrichtigt zu werden, wenn es notwendig wird, ist bei jedem vorhanden. Das Krisenmanagementsystem zeigt die Installation der erforderlichen physischen Geräte und die Programmierung des Krisenmanagementsystems.

In der Folge wird detailliert erklärt, welche Entwicklungsschritte zu setzen sind, um eine funktionstüchtige Anwendung zu erhalten. Jede/r kann entwickeln und das System somit selbst bauen, da die einzelnen Units einfach anzustecken sind und die Programmierung mit einfachen visuellen Symbolen und Code-Ausschnitten erfolgt.

Achtung: Es gibt zwei Versionen des Krisenmanagementsystems, einmal mit den Units Laser RX und TX und einmal mit der Licht Unit. Der Code und der M5Stack mit seinen zusammengesteckten Komponenten müssen demzufolge von derselben Version sein.

5.1. Systemarchitektur (M5 Stack)

Für die Bauanleitung verwenden wir Bilder, um die Reihenfolge zu erklären, wie die Units miteinander zu verbinden sind. Anschließend zeigen wir den gesamten Code, der benötigt wird, um eine funktionierende Anwendung zu erhalten.

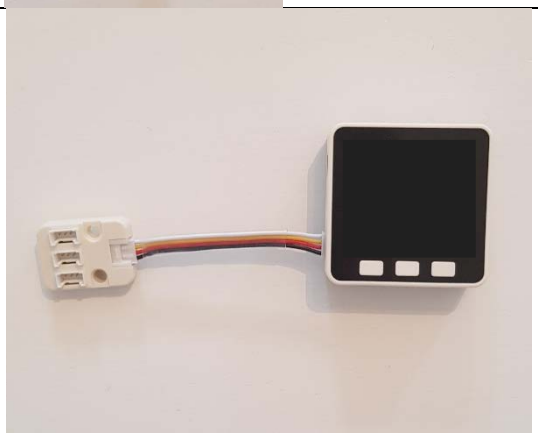
Um den Einstieg zu erleichtern, wird die erste Unit detaillierter beschrieben. Dies inkludiert,

- wo sie angesteckt werden muss,
- welcher Code-Block dafür verwendet wird und
- was dieser Code-Block leistet

Danach zeigt die Reihenfolge der Bilder, wie das Krisenmanagementsystem realisiert werden kann.

Um zu verstehen, welcher Code-Abschnitt für welche Unit notwendig ist, ist jeder Unit eine Zahl zugwiesen. Diese Zahl findet sich im Code-Abschnitt wieder und zeigt, welche Code-Zeile oder Code-Block mit der nummerierten Unit in Verbindung steht. Konnten alle Schritte erfolgreich bewältigt werden, sieht das Endergebnis wie in Abbildung 35 aus. Ausschlaggebend für ein erfolgreiches Erstellen des Krisenmanagementsystems ist, dass alle Schritte zur Vorbereitung, wie in der Checkliste angeführt, erfolgreich abgeschlossen werden konnten. In den einzelnen Überschriften ist angegeben, um welche Version es sich momentan handelt.

Abbildung 35: M5Stack mit den dazugehörigen Units für das Krisenmanagmentsystem

1		M5Stack einschalten und am Computersystem mit dem USB-C-Kabel anstecken. Danach in den USB Mode setzen.
2		Grove-Hub mit dazugehörigem Grove-Kabel bereitstellen.
2.a		Grove Kabel wird an den Grove-Hub angeschlossen.
2.b		Der Grove-Hub mit dem angeschlossenen Grove-Kabel, wird an Port A des M5Stack angeschlossen.

3		Den Environment Sensor mit dazugehörigem Grove-Kabel vorbereiten.
3.a		Das Grove-Kabel mit der Unit verbinden.
3.b		Den Environment Sensor mit dem angeschlossenen Grove- Kabel an den Grove-Hub anstecken.
4		Die Unit Pb.Hub und das Grove-Kabel vorbereiten.

4.a		Das Grove-Kabel an den Pb.Hub anschließen.
4.b		Den Pb.Hub mit dem angeschlossenen Grove-Kabel an den Grove-Hub anstecken.
Info	Mit dem Anschluss von einem Grove-Hub und einem Pb.Hub sind die erforderlichen Erweiterung der Port Anschlüsse für das Krisenmanagementsystem geschaffen. Jetzt kommen wir zur Unit Motion-Sensor. Diese Unit ermöglicht, Bewegung im Raum zu erkennen. Für unser Projekt genau die richtige Unit, um die Sicherheit zu erhöhen. Entsprechend der weiteren Anweisung ist nun der Motion-Sensor am M5Stack anzustecken.	

5		Die Unit Motion-Sensor mit dazugehörigem Grove-Kabel vorbereiten.
5.a		Das Grove-Kabel mit der Unit verbinden.
5.b		Die Motion-Sensor Unit mit dem angeschlossenen Grove-Kabel an den Pb.Hub anschließen. (**Achtung:** Es ist unbedingt notwendig, dass diese Unit an den Steckplatz 0 angeschlossen wird. Die Steckplätze sind alle auf der Unit nummeriert.)
Info	Nachdem der Motion-Sensor am Steckplatz 0 des Pb.Hubs angebracht wurde, folgt nun der Code-Block, um den Sensor nutzen zu können.	

Info

```
from m5stack import *
from m5ui import *
from uiflow import *
pbhub0 = unit.get(unit.PBHUB, unit.PORTA)
pirStatus = pbhub0.analogRead(0)

if pirStatus >= 1000:
  json = json.replace('@Pir@', 'bewegt sich etwas.')
else:
  json = json.replace('@Pir@', 'herrscht Ruhe.')
```

- Durch die “from…” Zeilen werden die benötigten Funktionen für das Programm geladen
- Unit.get() – Diese Funktion gibt dem Programm Bescheid, dass ein Pb.Hub angesteckt ist.
- analogRead() – Diese Funktion gibt Bescheid, an welchem Steckplatz der Motion Sensor angesteckt ist und speichert die Inhalte in die Variable „pirStatus“
- Mittels der If-Abfrage wird kontrolliert, ob eine Bewegung stattfindet oder nicht. Wenn der Wert über 1000 ist, wurde eine Bewegung erkannt. Sonst nicht.

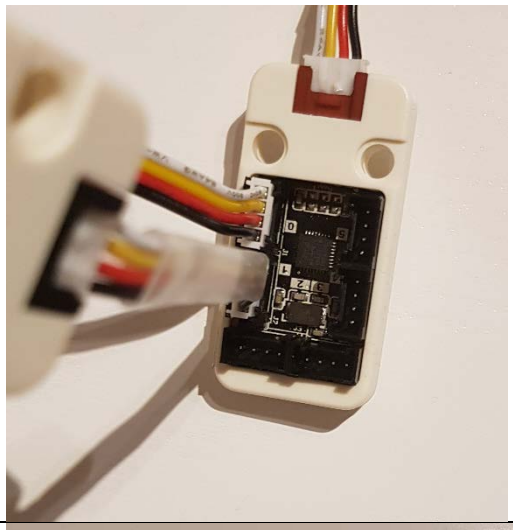
6		Bislang wurden der Grove-Hub, der Pb.Hub und der Motion-Sensor verwendet.
7		Einen weiteren Grove-Hub mit einem Grove-Kabel vorbereiten.

7.a		Grove-Hub mit dem Grove-Kabel verbinden.
8		Der Grove-Hub mit dem angeschlossenen Grove-Kabel wird an Port B des M5Stack angeschlossen.
Info	Die Version 1 (Der Einsatz von Laser RX und TX Unit, wird nun gezeigt)	

9		Laser RX Unit und Laser TX Unit mit dazugehörigen Grove-Kabeln vorbereiten.
9.a		Laser RX Unit mit einem Grove-Kabel verbinden und Laser TX Unit mit dem anderen Grove-Kabel verbinden.
10		Laser TX Unit und Laser RX Unit mit dem angeschlossenen Grove-Kabel an den Grove-Hub anschließen. Aufnahme von den aktuell angeschlossenen Units und dem M5Stack, nachdem die beiden Units Laser TX und Laser RX hinzugefügt wurden.

Info		Die Version 2 (Der Einsatz vom Light Modul statt den Modulen Laser RX und TX, wird nun gezeigt). Hierfür ist ein Einstieg bei Schritt 6 notwendig (dargestellt in der Abbildung links), und von dort aus wird nun weitergebaut.
11		Light Unit mit dazugehörigem Grove-Kabel vorbereiten.
11.a		Light Unit mit einem Grove-Kabel verbinden.

11.b



Die Light Unit mit dem angeschlossenen Grove-Kabel an den Pb.Hub anschließen. (Achtung: Es ist unbedingt notwendig, dass diese Unit an den Steckplatz 1 angeschlossen wird. Die Steckplätze sind alle auf der Unit nummeriert.)

12

Aufnahme von den aktuell angeschlossenen Units und dem M5Stack, nachdem die Light Unit hinzugefügt wurden

5.2. Programm - Python Code (Version 1)

Nachfolgend ist der Programm Code für die 1. Version eines Krisenmanagementsystems in der Programmiersprache Python zu sehen. Um diesen zu verwenden, muss man den nachfolgenden Code in UI-Flow hineinkopieren. Dabei ist darauf zu achten, dass man den Karteireiter Python in UI-Flow ausgewählt hat und nicht Blockly. Beim Kopieren werden die Info Boxen (in grüner Schrift), die geschwungenen Klammern und die Zahlen, welche als Verweis auf das zu verwendende Modul in dieser Anleitung dienen, nicht mitkopiert, diese sind lediglich zur Information angeführt, beinhalten jedoch keine Funktion.

```
from m5stack import *
from m5ui import *
from uiflow import *
import unit
import random
import unit
import wifiCfg
import socket
import _thread as th
```

Info: Hiermit werden benötigte Ressourcen geladen, um später eine WiFi Verbindung herzustellen oder mit dem M5Stack zu kommunizieren.

```
setScreenColor(0x000000)
```

```
labelWebServer = M5TextBox(240, 20, "WS",
lcd.FONT_Default,0xFFFFFF, rotate=0)
labelIpAddress = M5TextBox(300, 170,
"xxx.xxx.xxx.xxx", lcd.FONT_Default,0xFFFFFF,
rotate=270)
levelPercent = 100
```

Info: Erstellt zwei Labels, um die IP- Adresse auf dem M5Stack Bildschirm auszugeben.

```
ButtonInput = None
```

```
#setting pbHub
pbhub0 = unit.get(unit.PBHUB, unit.PORTA)
```

④ **Info**: Der Pb.Hub wird hiermit vom M5Stack erkannt.

```
#setting Variables for Environment Modul
env0 = unit.get(unit.ENV, unit.PORTA)
envTemp = None
envPress = None
```

③ **Info**: Der Environment Sensor wird hiermit vom M5Stack erkannt.

```
envHum = None

#setting Variables for Motion Sensor
pir0 = unit.get(unit.PIR, unit.PORTB)
pirStatus = None
```

⑤ **Info**: Der Motion Sensor wird hiermit vom M5Stack erkannt.

```
#setting Laser RX and Laser TX
uart = None
uartRead = None
laserRead = None
laserValue = None
global i
i = 0
uart=machine.UART(1,rx=36,tx=26,timeout=50)
uart.init(9600, bits=8, parity=0, stop=1)
```

⑤ **Info**: Laser RX und Laser TX werden hiermit vom M5Stack erkannt.

```
wifiCfg.screenShow()
wifiCfg.autoConnect(lcdShow = True)

def buttonA_wasPressed():
  global ButtonInput
  ButtonInput = 'A'
  pass
btnA.wasPressed(buttonA_wasPressed)
```

Info: Methode, um dem A Knopf drücken zu können und diese Information weiterzuleiten.

```
def buttonB_wasPressed():
  global ButtonInput
  ButtonInput = 'B'
  pass
btnB.wasPressed(buttonB_wasPressed)
```

Info: Methode, um dem B Knopf drücken zu können und diese Information weiterzuleiten.

```
def buttonC_wasPressed():
  global ButtonInput
```

Info: Methode, um dem C Knopf drücken zu können und diese Information weiterzuleiten.

```
  ButtonInput = 'C'
  pass
btnC.wasPressed(buttonC_wasPressed)


def webServer():


    global levelPercent, setupMode


    web_response_json =
"""{"data":[{"deviceName":"Server","deviceId":
"M5Stack Input","value": "Button
Pressed:","ButtonPressValue":
"@ButtonValue@"},
    {"deviceName":"Environment
Unit:","deviceId":"Temperature","unit":
"Temperature:", "value": "@Temp@"},
    {"deviceName":"Environment
Unit:","deviceId":"Pressure","unit":
"Pressure:", "value": "@Press@"},
    {"deviceName":"Environment
Unit:","deviceId":"Humidity", "unit":
"Humidity:","value": "@Hum@"},
    {"deviceName":"Motion
Unit:","deviceId":"@Pir@", "unit":
"Bewegung","value": "Im aktuellen Raum"},
    {"deviceName":"Laser
Unit:","deviceId":"@Laser@", "unit":
"@LaserValue@","value": "Die Schranke"}
    ]}"""


    web_error_404 = """<!DOCTYPE HTML PUBLIC
"-//IETF//DTD HTML
2.0//EN"><html><head><title>404 Not
Found</title></head><body><h1>Not
Found</h1><p>The requested URL was not found
on this server.</p></body></html>"""


    while True:


        try:
```

Info: Methode, um den Webserver zu erstellen. Definiert zusätzlich die Ausgabe in einem JSON Format.

```
            labelIpAddress.setText('')
            i = uart.any()
            if (i > 0):
              uartRead = str(uart.read())
              laserRead ='wurde durchbrochen'
              laserValue = 1
            if (i <= 0):
              uartRead = str(uart.read())
              laserRead = 'wurde nicht
durchbrochen'
              laserValue = 0
            envTemp = (str(env0.temperature))
            envPress = (str(env0.pressure))
            envHum = (str(env0.humidity))
            pirStatus = pbhub0.analogRead(0)


            if not
wifiCfg.wlan_sta.isconnected():
                labelWebServer.setText('WS:
RECON')
                wifiCfg.reconnect()
            else:
                ip =
wifiCfg.wlan_sta.ifconfig()
                labelIpAddress.setText('IP ' +
wifiCfg.wlan_sta.ifconfig()[0])


                server =
socket.socket(socket.AF_INET,
socket.SOCK_STREAM)


server.setsockopt(socket.SOL_SOCKET,
socket.SO_REUSEADDR, 1)
                server.settimeout(30)
                server.bind((ip[0], 80))
```

Info: envTemp erkennt die Temperatur
envPress erkennt den Luftdruck
envHum erkennt die Luftfeuchtigkeit
Info: pirStatus gibt die Information weiter, wenn eine Bewegung vom Bewegungssensor erkannt wird

```
                server.listen(1)

                labelWebServer.setText('WS:
RUNS')

                (conn, addr) = server.accept()
                request = conn.recv(1024)

                if request:
                  request = str(request)

                  isRequestLevel =
request.find('/api/v1.0/level ') >= 0

                  if isRequestLevel:

labelWebServer.setText('WS: 200')
                      json = web_response_json
                      json =
json.replace('@value@',
str('%.2f'%levelPercent))
                      json =
json.replace('@ButtonValue@',
str(ButtonInput))
                      json =
json.replace('@Temp@', str(envTemp))
                      json =
json.replace('@Press@', str(envPress))
                      json =
json.replace('@Hum@', str(envHum))
                      if pirStatus >= 1000:
                        json =
json.replace('@Pir@', 'bewegt sich etwas.')
                      else:
                        json =
json.replace('@Pir@', 'herrscht Ruhe.')
                      json =
json.replace('@Laser@', str(laserRead))
```

Info: json.replace() = ersetzt den Wert zwischen den zwei „@“ Symbolen durch die Variable die nach dem Beistrich folgt.

Info: if pirStatus >= 1000 -> Wenn der Wert über 1000 ist, wurde eine Bewegung erkannt, sonst nicht.

```
                        json =
json.replace('@LaserValue@', str(laserValue))
                        response = json
                        conn.send('HTTP/1.1 200
OK\r\n')
                        conn.send('Content-Type:
application/json\r\n')
                        conn.send('Connection:
close\r\n\r\n')
                    else:

labelWebServer.setText('WS: 404')
                        response = web_error_404
                        conn.send('HTTP/1.1 404
NOT FOUND\r\n')
                        conn.send('Content-Type:
text/html\r\n')
                        conn.send('Connection:
close\r\n\r\n')


                    conn.sendall(response)


        except Exception as e:
            gc.collect()
            labelWebServer.setText('WS: ERR')


        finally:
            try:
                conn.close()
                server.close()
            except Exception as e:
                pass
            finally:
                pass
th.start_new_thread(webServer, ())
```

5.3. Programm - Python Code (Version 2)

Nachfolgend ist der Programm Code für die 2. Version eines Krisenmanagementsystems in der Programmiersprache Python zu sehen. Um diesen zu verwenden, muss man den nachfolgenden Code in UI-Flow hineinkopieren. Dabei ist darauf zu achten, dass man den Karteireiter Python in UI-Flow ausgewählt hat und nicht Blockly. Beim Kopieren werden die Info Boxen (in grüner Schrift), die geschwungenen Klammern und die Zahlen, welche als Verweis auf das zu verwendende Modul in dieser Anleitung dienen, nicht mitkopiert, diese sind lediglich zur Information angeführt, beinhalten jedoch keine Funktion.

```
from m5stack import *
from m5ui import *
from uiflow import *
import unit
import random
import unit
import wifiCfg
import socket
import _thread as th
```

Info: Hiermit werden benötigte Ressourcen geladen, um später eine WiFi Verbindung herzustellen oder mit dem M5Stack zu kommunizieren.

```
setScreenColor(0x000000)
```

```
labelWebServer = M5TextBox(240, 20, "WS",
lcd.FONT_Default,0xFFFFFF, rotate=0)
labelIpAddress = M5TextBox(300, 170,
"xxx.xxx.xxx.xxx", lcd.FONT_Default,0xFFFFFF,
rotate=270)
levelPercent = 100
ButtonInput = None
```

Info: Erstellt zwei Labels, um die IP-Adresse auszugeben auf dem M5Stack Bildschirm.

```
#setting pbHub
pbhub0 = unit.get(unit.PBHUB, unit.PORTA)
```

④ **Info**: Der Pb.Hub wird hiermit vom M5Stack erkannt.

```
#setting Variables for Environment Modul
env0 = unit.get(unit.ENV, unit.PORTA)
envTemp = None
envPress = None
envHum = None
```

③ **Info**: Der Environment Sensor wird hiermit vom M5Stack erkannt.

```
#setting Variables for Motion Sensor
pir0 = unit.get(unit.PIR, unit.PORTB)
pirStatus = None
```

⑤ **Info**: Der Motion Sensor wird hiermit vom M5Stack erkannt.

```
#setting Variables for Light Sensor
light 0 = unit.get(unit.LIGHT, unit.PORTB)
lightStatus = None
```

⑪ **Info**: Light Unit wird hiermit vom M5Stack erkannt.

```
wifiCfg.screenShow()
wifiCfg.autoConnect(lcdShow = True)
```

```
def buttonA_wasPressed():
  global ButtonInput
  ButtonInput = 'A'
  pass
btnA.wasPressed(buttonA_wasPressed)
```

Info: Methode, um den A Knopf drücken zu können und diese Information weiterzuleiten.

```
def buttonB_wasPressed():
  global ButtonInput
  ButtonInput = 'B'
  pass
btnB.wasPressed(buttonB_wasPressed)
```

Info: Methode, um den B Knopf drücken zu können und diese Information weiterzuleiten.

```
def buttonC_wasPressed():
  global ButtonInput
  ButtonInput = 'C'
  pass
btnC.wasPressed(buttonC_wasPressed)
```

Info: Methode, um dem C Knopf drücken zu können und diese Information weiterzuleiten.

```
def webServer():
```

Info: Methode, um den Webserver zu erstellen. Definiert zusätzlich die

```
    global levelPercent, setupMode


    web_response_json =
"""{"data":[{"deviceName":"BlablaLevel","deviceId":
"M5Stack Input","value": "Button
Pressed:","ButtonPressValue": "@ButtonValue@"},

    {"deviceName":"Environment
Unit:","deviceId":"Temperature","unit":
"Temperature:", "value": "@Temp@"},

    {"deviceName":"Environment
Unit:","deviceId":"Pressure","unit": "Pressure:",
"value": "@Press@"},

    {"deviceName":"Environment
Unit:","deviceId":"Humidity", "unit":
"Humidity:","value": "@Hum@"},

    {"deviceName":"Motion
Unit:","deviceId":"@Pir@", "unit":
"Bewegung","value": "Im aktuellen Raum"},

    {"deviceName":"Light
Unit:","deviceId":"@Light@", "unit":
"@LightValue@","value": "Licht:"}

    ]}"""

    web_error_404 = """<!DOCTYPE HTML PUBLIC "-
//IETF//DTD HTML 2.0//EN"><html><head><title>404
Not Found</title></head><body><h1>Not
Found</h1><p>The requested URL was not found on
this serv er.</p></body></html>"""


    while True:


        try:


            labelIpAddress.setText('')

            envTemp = (str(env0.temperature))

            envPress = (str(env0.pressure))

            envHum = (str(env0.humidity))

            pirStatus = pbhub0.analogRead(0)

            lightStatus = pbhub0.analogRead(1)
```

Ausgabe in einem JSON Format.

Info: envTemp erkennt die Temperatur
envPress erkennt den Luftdruck
envHum erkennt die Luftfeuchtigkeit
Info: pirStatus = erkennt Bewegungen

```
            if not wifiCfg.wlan_sta.isconnected():
                labelWebServer.setText('WS: RECON')
                wifiCfg.reconnect()
            else:
                ip = wifiCfg.wlan_sta.ifconfig()
                labelIpAddress.setText('IP ' +
wifiCfg.wlan_sta.ifconfig()[0])


                server =
socket.socket(socket.AF_INET, socket.SOCK_STREAM)

server.setsockopt(socket.SOL_SOCKET,
socket.SO_REUSEADDR, 1)
                server.settimeout(30)
                server.bind((ip[0], 80))
                server.listen(1)


                labelWebServer.setText('WS: RUNS')


                (conn, addr) = server.accept()
                request = conn.recv(1024)


                if request:
                  request = str(request)


                  isRequestLevel =
request.find('/api/v1.0/level ') >= 0


                  if isRequestLevel:
                      labelWebServer.setText('WS:
200')
                      json = web_response_json
                      json =
json.replace('@value@', str('%.2f'%levelPercent))
                      json =
json.replace('@ButtonValue@', str(ButtonInput))
```

Info: lightStatus = erkennt Licht im Raum

Info: json.replace() = ersetzt den Wert zwischen den zwei „@“ Symbolen

```
                        json = json.replace('@Temp@', str(envTemp))
                        json = json.replace('@Press@', str(envPress))
                        json = json.replace('@Hum@', str(envHum))
                        if lightStatus > 700:
                          json = json.replace('@Light@', 'Ausgeschaltet')
                          json = json.replace('@LightValue@', str(0))
                        else:
                          json = json.replace('@Light@', 'Eingeschaltet')
                          json = json.replace('@LightValue@', str(1))
                        if pirStatus >= 1000:
                          json = json.replace('@Pir@', 'bewegt sich etwas.')
                        else:
                          json = json.replace('@Pir@', 'herrscht Ruhe.')
                        response = json
                        conn.send('HTTP/1.1 200 OK\r\n')
                        conn.send('Content-Type: application/json\r\n')
                        conn.send('Connection: close\r\n\r\n')
                    else:
                        labelWebServer.setText('WS: 404')
                        response = web_error_404
                        conn.send('HTTP/1.1 404 NOT FOUND\r\n')
                        conn.send('Content-Type: text/html\r\n')
                        conn.send('Connection: close\r\n\r\n')
```

durch die Variable die nach dem Beistrich folgt

Info: if lightStatus >= 700 -> Wenn der Wert über 700 ist, wurde Licht im Raum erkannt, sonst nicht.

Info: if pirStatus >= 1000 -> Wenn der Wert über 1000 ist, wurde eine Bewegung erkannt, sonst nicht.

```
                conn.sendall(response)

        except Exception as e:
            gc.collect()
            labelWebServer.setText('WS: ERR')

        finally:
            try:
                conn.close()
                server.close()
            except Exception as e:
                pass
            finally:
                pass


th.start_new_thread(webServer, ())
```

5.4. Anwendungsfall-Logik (App Inventor)

Sobald alle Units korrekt angesteckt wurden und der Python Code in UIFlow hinzugefügt wurde, geht es im nächsten Schritt um die Erstellung der mobilen Applikation.

Zuerst müssen die benötigten Komponenten in MIT App Inventor hinzugefügt werden. Wie im Abschnitt MIT App Inventor erwähnt, geht man dafür auf die angezeigten Komponenten im Paletten Bereich, um eine Auflistung aller vorhandenen Komponenten zu erhalten. Für Version 1 und Version 2 ist der Aufbau der mobilen Applikation gleich.

In der nächsten Abbildung wird eine Auflistung aller Komponenten vom „User Interface“ angezeigt sowie die anderen Metakategorien.

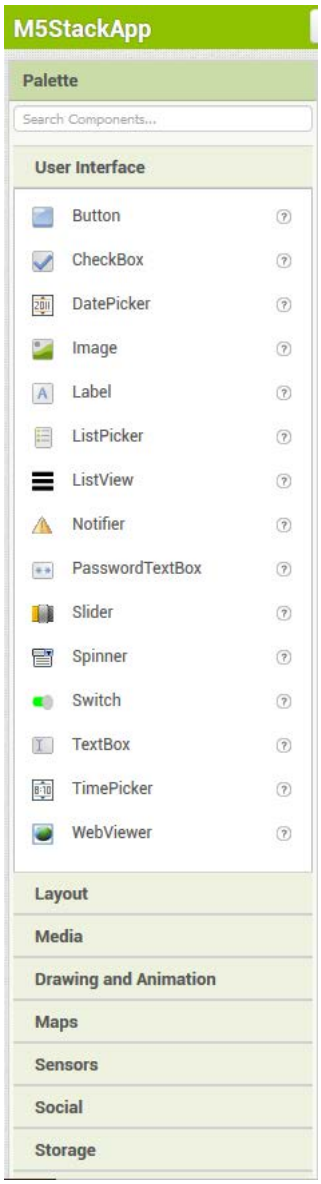


Abbildung 36: Ansicht aller Komponenten unter "Palette"

Anschließend wählt man die gewünschte Komponente aus und fügt diese durch Drag-and-Drop auf die Benutzeroberfläche hinzu. Für das Krisenmanagementsystem empfiehlt es sich, mit dem

„HorizontalArrangement“ im „Layout“ Bereich zu beginnen. Allerdings spielt die Reihenfolge, in der die Komponenten hinzugefügt werden, keine Rolle.

In der nächsten Abbildung werden alle benötigten Komponenten aufgezeigt, die notwendig sind für das Krisenmanagementsystem.

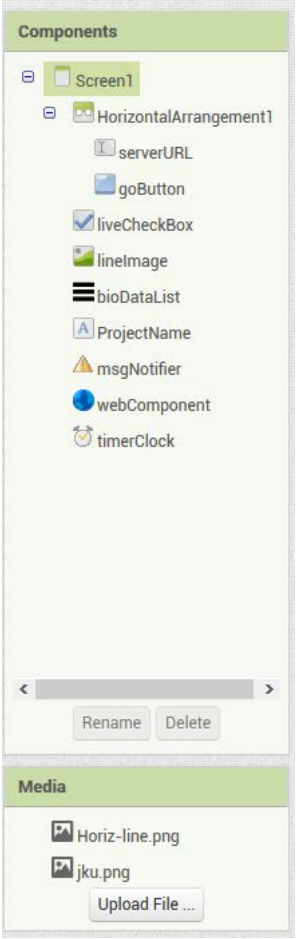


Abbildung 37: Benötigte Komponenten, um die mobile Applikation zu erstellen

Wie bereits im Abschnitt MIT App Inventor erklärt, wird eine Komponente per Drag-and-Drop auf die Oberfläche gezogen. Danach ist es notwendig, der neu erstellten Komponente einen passenden Namen zu geben, um diese später ansprechen zu können. In den nächsten Seiten werden zudem die notwendigen Eigenschaften (Properties) von den benötigten Komponenten aufgezeigt. Damit wird sichergestellt, dass alle Einstellungen korrekt vorgenommen werden und die mobile Applikation am Ende erfolgreich ausgeführt wird.

Die erste Komponente, die hinzugefügt wird, ist „HorizontalArrangement“. Die Eigenschaften sowie der Name der Komponente wird in der nächsten Abbildung dargestellt.

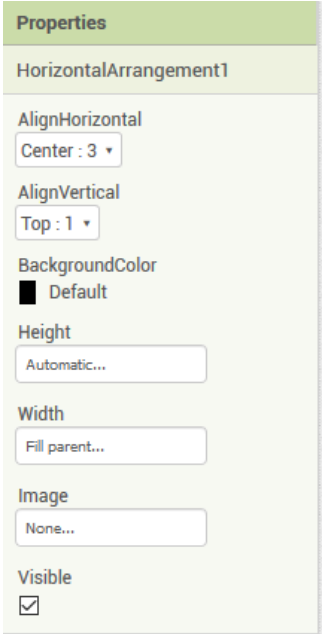


Abbildung 38: Einstellung von HorizontalArrangement1

Anschließend wählen wir die Texting Komponente unter im Panel auf der linken Seite unter „Social“ aus und ziehen diese Komponente auf die bereits existierende Komponente „HorizontalArrangement1“. Hierbei ist es wichtig zu gewährleisten, dass die Texting Komponente korrekt abgelegt wird. Diese ist dann korrekt abgelegt, wenn in der Ansicht die neue Komponente etwas Rechts eingerückt dargestellt wird wie in der nächsten Abbildung zu sehen ist.

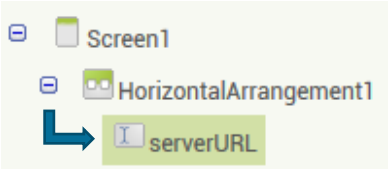


Abbildung 39: Die serverURL Komponente muss unter HorizontalArrangement1 stehen

Nachdem die Komponente erfolgreich hinzugefügt wurde, erhält diese den Namen „serverURL“. Die Eigenschaften werden in der darunter stehenden Graphik angezeigt.

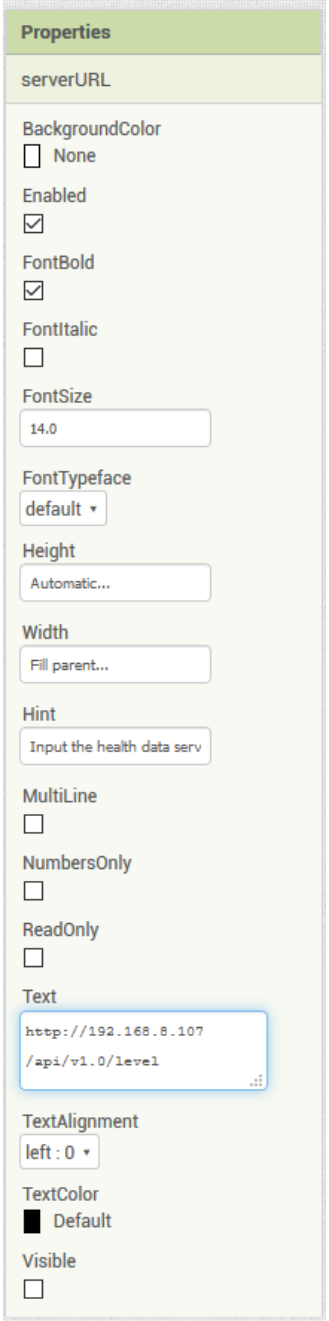


Abbildung 40: Die Eigenschaften von der serverURL Komponente

Hierbei ist anzumerken, dass als „Text“ die IP-Adresse eingegeben werden soll, die vom M5Stack erstellt wird. Diese wird am LCD-Display vom M5Stack angezeigt. Ein Beispiel könnte folgend aussehen: http://192.168.8.107/api/v1.0/level. Um mit der IP-Adresse weiter arbeiten zu können, ist es wichtig, dass sich der M5Stack im selben Netzwerk befindet, wie das Endgerät für die App. Wie der M5Stack mit einem bestimmten Netzwerk verbunden wird, ist in Kapitel 4.1 zu sehen.

Eine erfolgreiche IP-Verbindung wird in Abbildung 41 gezeigt, der Pfeil signalisiert, dass die Linke IP-Adresse verwendet werden soll.

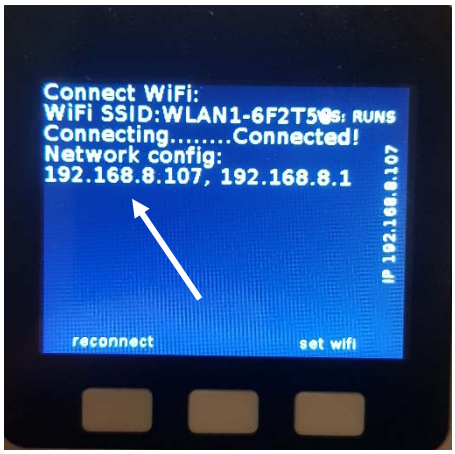


Abbildung 41: Die erstellte IP-Adresse vom M5Stack wird unter dem Text „Network Config“ angezeigt.

Ähnlich wie zur Texting Komponente, wird ein neuer Button namens „goButton“ hinzufügt. In Abbildung 42 werden die Eigenschaften dargestellt, die der „goButton“ aufweisen soll.

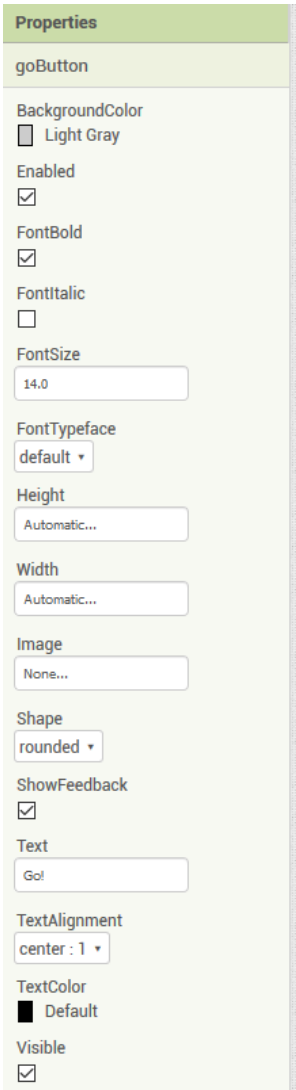


Abbildung 42: Die Eigenschaften von der goButton Komponente

Die nächste Komponente ist eine CheckBox namens „liveCheckBox“ mit folgenden Eigenschaften.

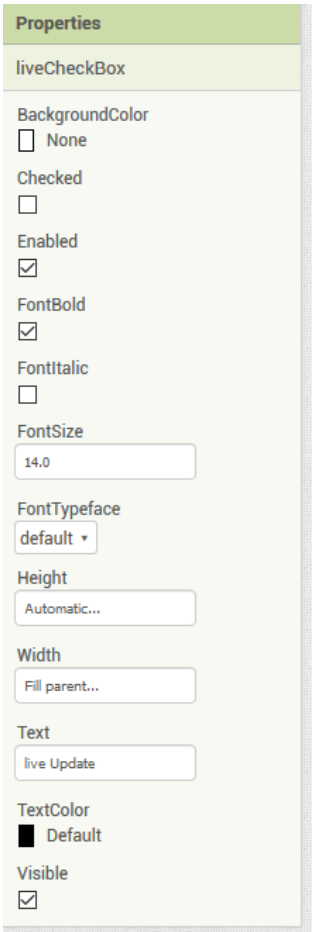


Abbildung 43: Die Eigenschaften von der liveCheckBox

Diese Komponente ermöglicht eine Auswahl zur durchgehenden (wiederholten) Datenübertragung vom M5Stack zur App oder einer einmaligen.

Die nächste Komponente ist optional, da sie nur optisch einen Vorteil bewirkt. Wir fügen ein Bild hinzu, um die Übersicht später zu verbessern namens „lineImage“. Wie unter Media bereits erwähnt, fügen wir ein Bild hinzu. Hierbei handelt es sich um einen kleinen dünnen Strich, wie im unteren Bild zu sehen ist.

Abbildung 44: Das hochgeladene Bild namens "lineImage"

Die Eigenschaften dieser Komponente sind in der nächsten Graphik zu sehen.

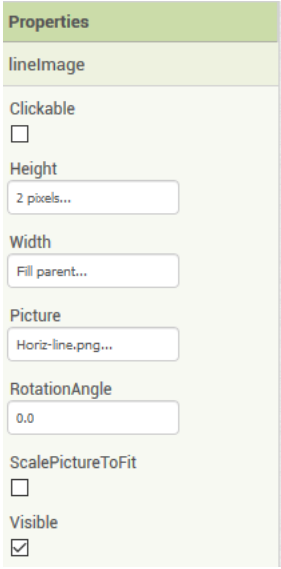


Abbildung 45: Die Eigenschaften der Komponente lineImage

Als nächstes fügen wir eine ListView Komponente namens „bioDataList“ hinzu. Durch diese Liste werden die Informationen vom M5Stack gespeichert und für uns lesbar ausgegeben. Die Eigenschaften werden in der unteren Abbildung gezeigt.

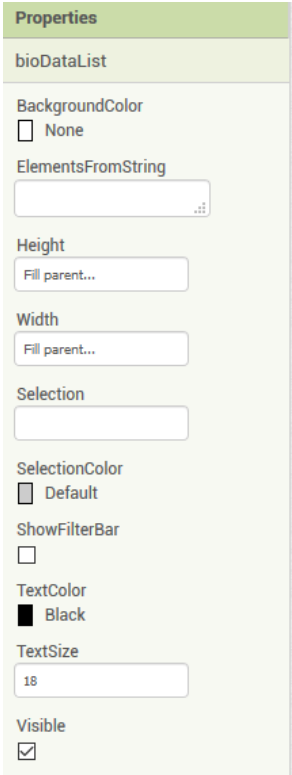


Abbildung 46: Die Eigenschaften der bioDataList Komponente

Die nächste Komponente ist ein Label namens „ProjectName“. Die Eigenschaften werden wieder in der nächsten Abbildung angezeigt.

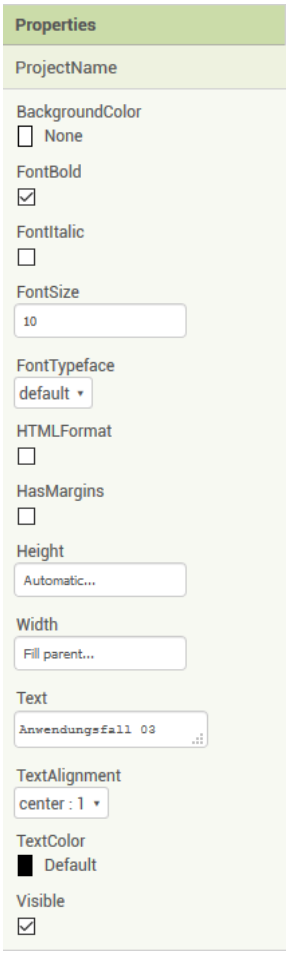


Abbildung 47: Die Eigenschaften der ProjectName Komponente

Eine weitere wichtige Komponente ist ein Notifier, namens „msgNotifier“ mit folgenden Eigenschaften.

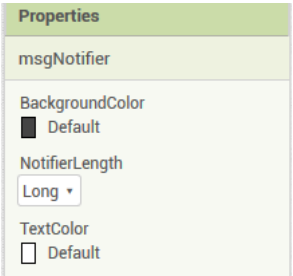


Abbildung 48: Die Eigenschaften der msgNotifier Komponente

Die vorletzte Komponente ist eine Web Komponente namens „webComponent“. Diese ermöglicht der Applikation eine Internetverbindung aufzubauen. Die Eigenschaften dazu sind in Abbildung 49 dargestellt.

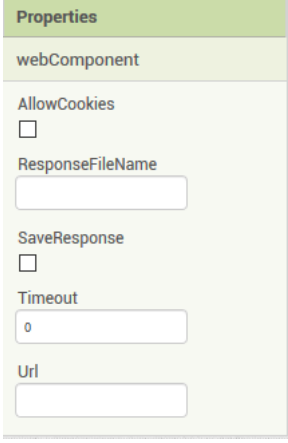


Abbildung 49: Die Eigenschaften der webComponent Komponente

Die letzte Komponente ist eine Clock namens „timerClock“ mit folgenden Eigenschaften.

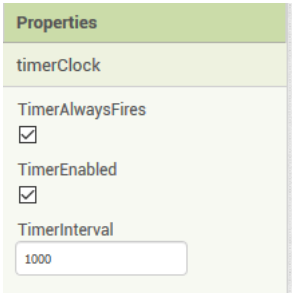


Abbildung 50: Die Eigenschaften der timerClock Komponente

5.5. Programm – App Inventor Code

Nachdem nun die Oberfläche der App kreiert wurde, muss die Funktionalität der App noch erstellt werden. Dafür müssen alle nachfolgenden Blöcke nachgebaut werden. Die Farben der Blöcke sind ein Hinweis darauf, in welcher Meta-Kategorie dieser Block zu finden ist.

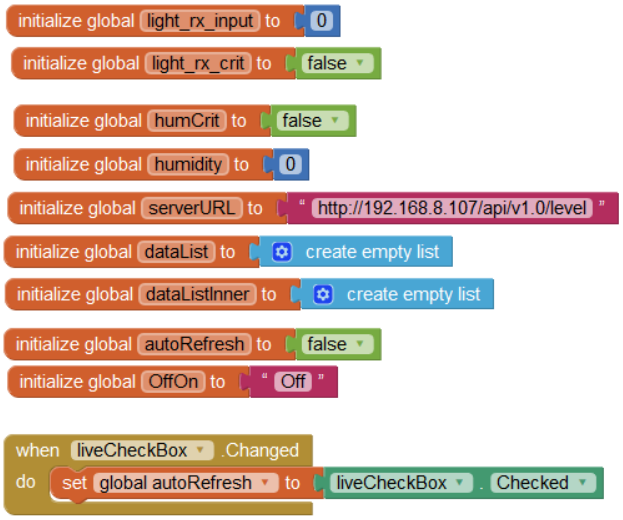


Abbildung 51: Die benötigten Variablen und die Methode für die live Check Box werden erstellt, wenn diese aktiv ist

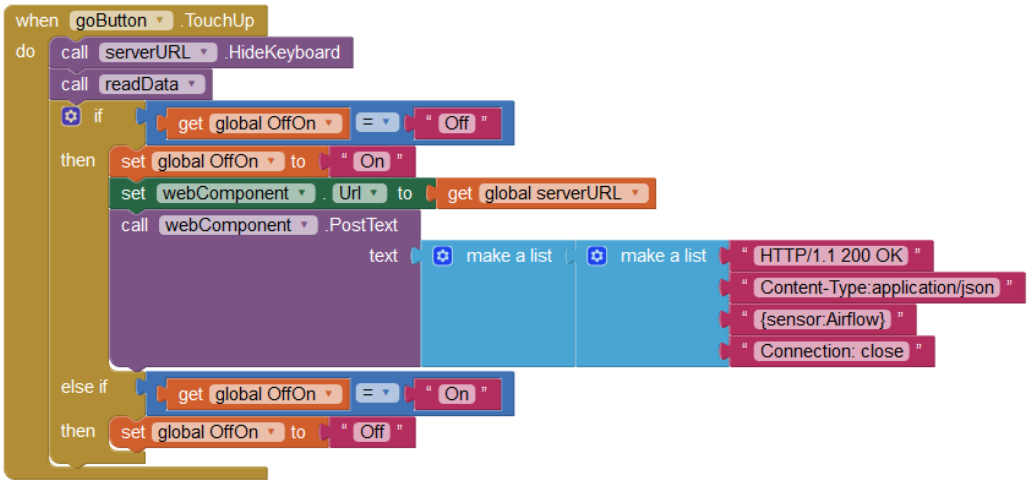


Abbildung 522: Die Methode, wenn der Go! Button gedrückt wird

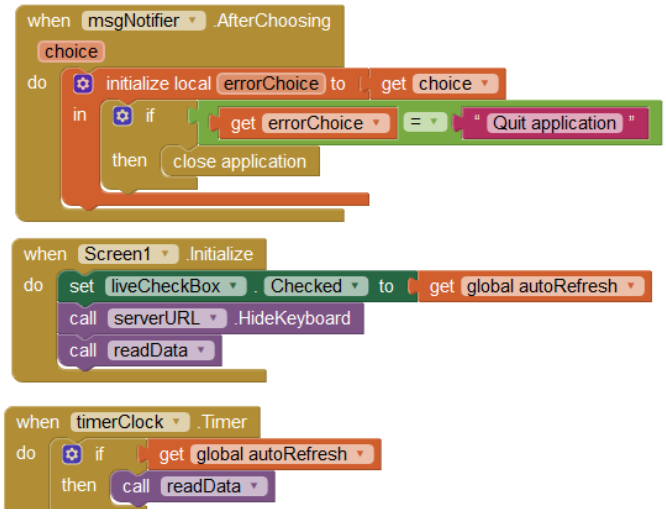


Abbildung 53: Drei verschiedene Methoden

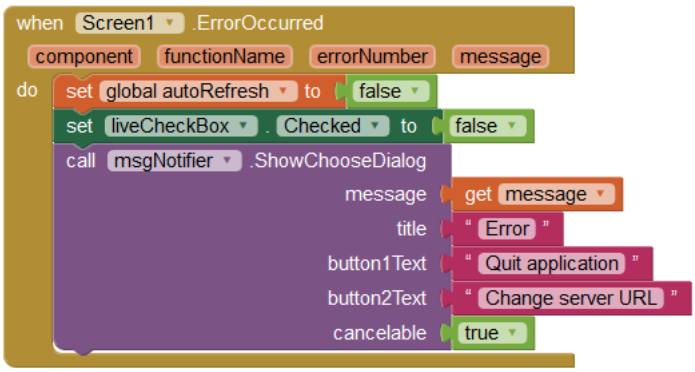


Abbildung 53: Innerhalb der Methode wird der App angewiesen, wie sie während der Laufzeit auf Fehler reagieren soll

In der nächsten Abbildung wird die readData() Methode gezeigt, welche als einzige einen Unterschied aufweist, je nachdem welche Version verwendet wird. Für die Version 1 muss nichts verändert werden. Für die Version 2 ist es notwendig, den Text mit „The light in your room was turned on“ mit dem Text „The barrier was broken.“ zu ersetzen. Dafür wird der Text per Drag and Drop verschoben.

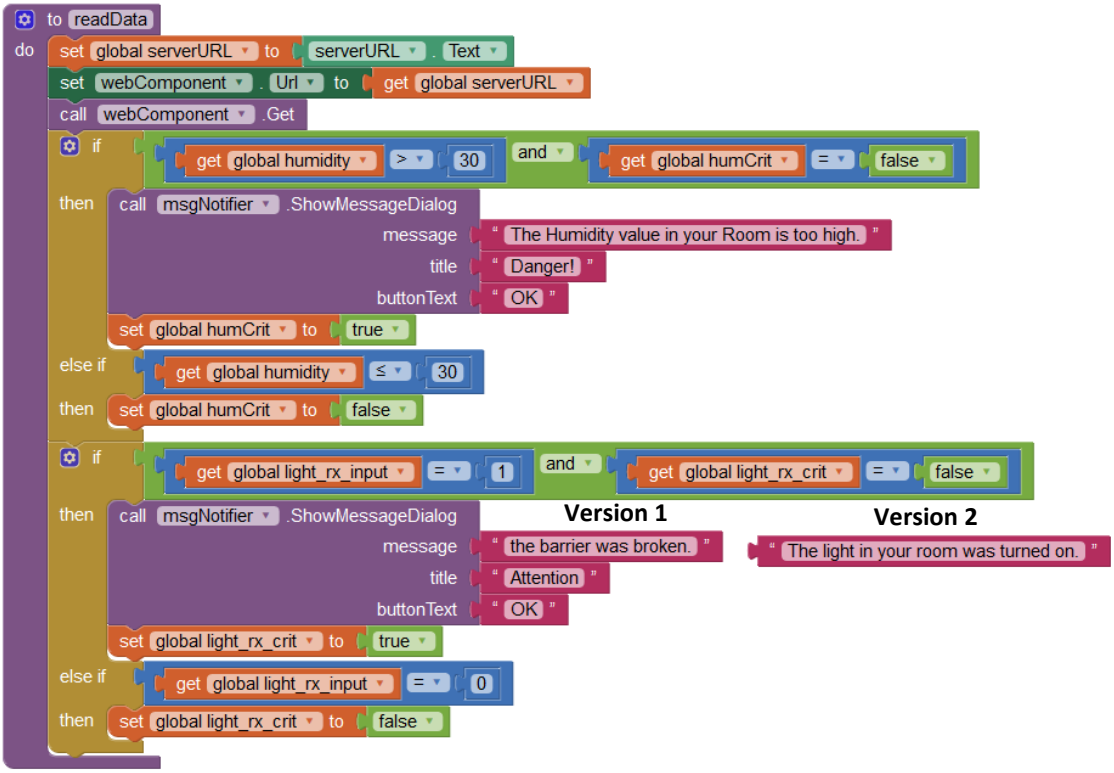


Abbildung 54: die readData Methode, welche Daten vom M5Stack erhält

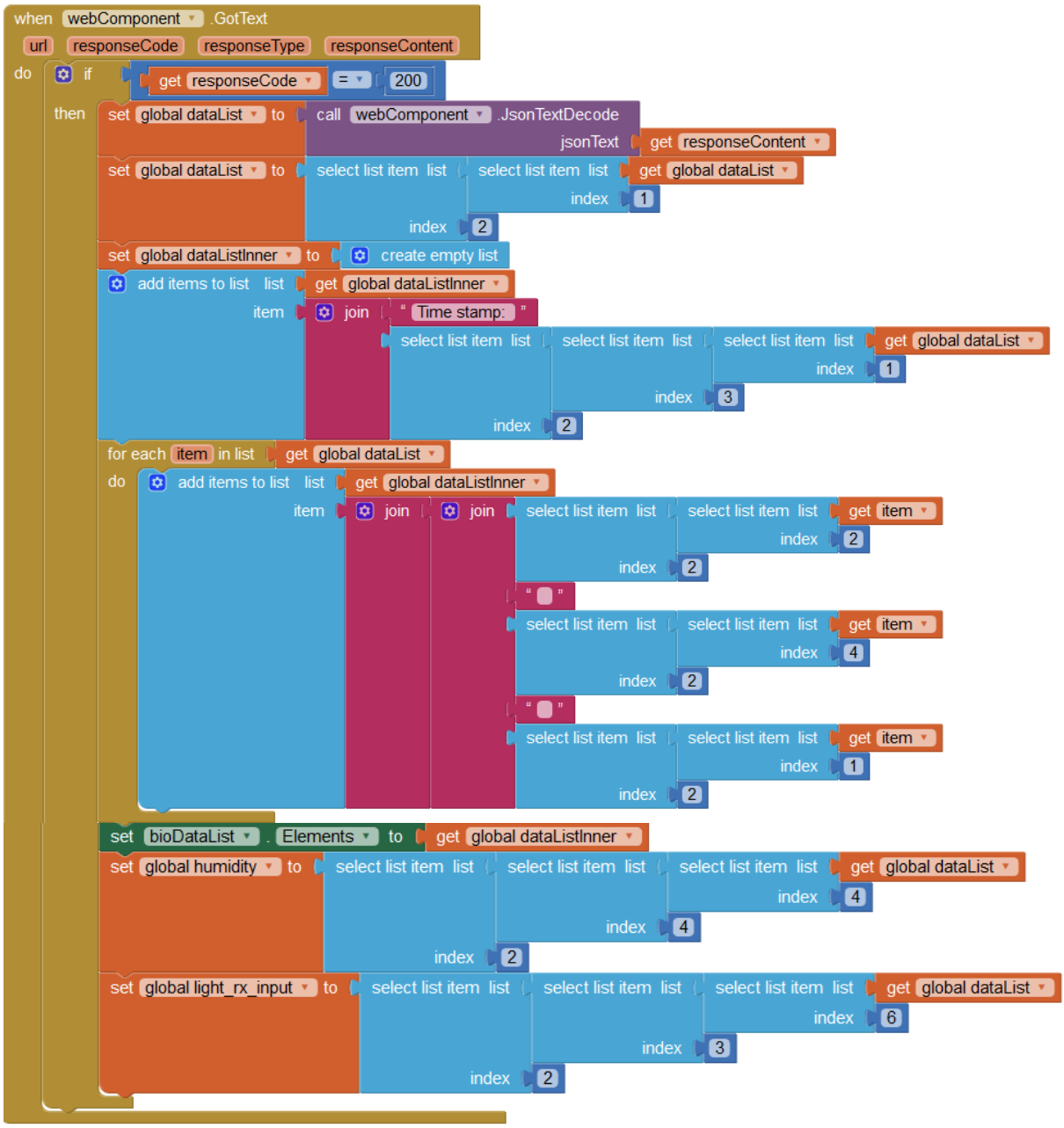


Abbildung 55: Die webComponent Methode, welche die erhaltenen Daten in eine Liste speichert

5.6. Mobile Applikation – Überblick

Das Krisenmanagementsystem weist die im Abschnitt Vorbereitung für das Programm beschriebenen Komponenten, Units und Codes auf. Eine Beschreibung wie die Applikation funktioniert wird in der nächsten Abbildung gezeigt.

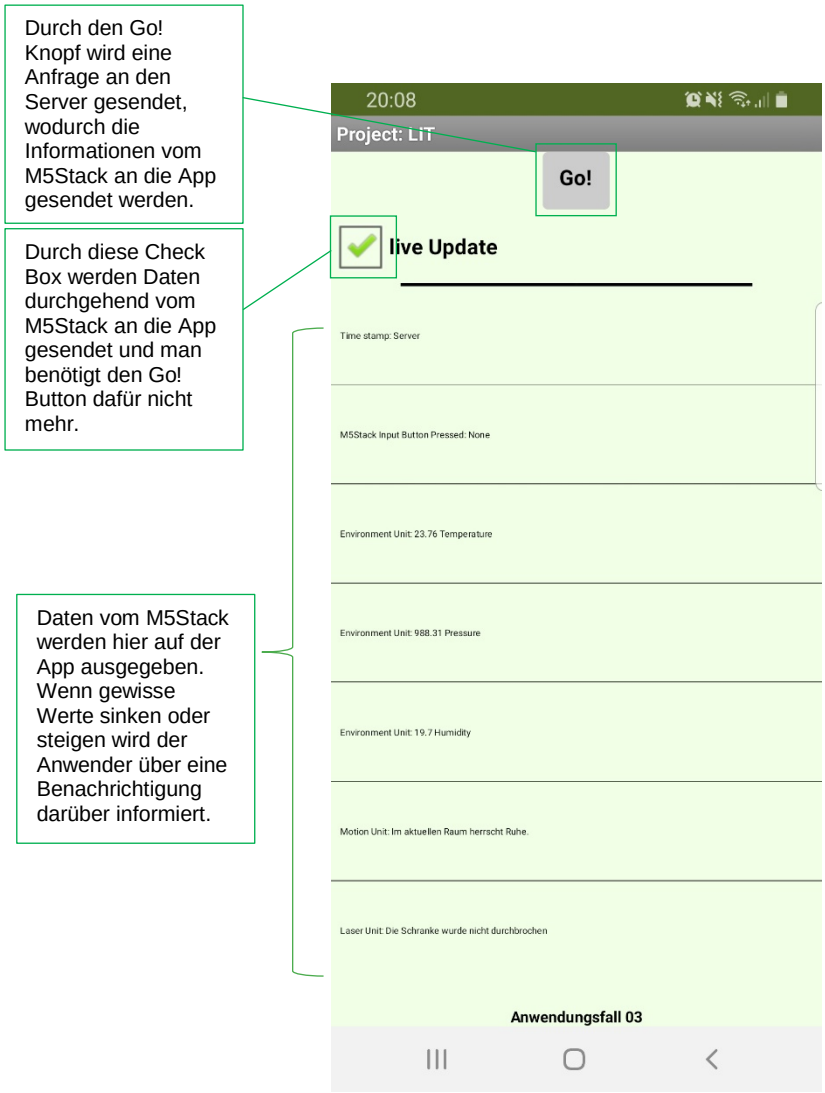
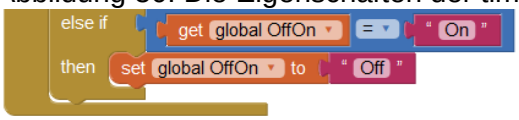


Abbildung 56: Ansicht und Erklärung von der mobilen Applikation

6. Abbildungsverzeichnis

Abbildung 1: M5Go IoT Starter Kit ESP32 5
Abbildung 2: Grove Hub Unit Abbildung 6
Abbildung 3: Pb.Hub Unit Abbildung 6
Abbildung 4: Motion Sensor Unit Abbildung 7
Abbildung 5: Environment Sensor Unit 7
Abbildung 6: Laser RX Unit Abbildung 7
Abbildung 7: Laser TX Unit Abbildung 8
Abbildung 8: Optische Darstellung der mobilen Applikation 8
Abbildung 9: M5Stack und die dazugehörigen Units für das Krisenmanagmentsystem 8
Abbildung 10: Beschriftung vom M5Stack 10
Abbildung 11: Ordnerstruktur vom M5Burner 10
Abbildung 12: Benutzeroberfläche vom M5Burner 11
Abbildung 13: Setting Auswahl vom UIFlow 12
Abbildung 14: Die Ansicht, nachdem ein Rechtsklick auf das Windows Symbol ausgeführt wurde 13
Abbildung 15: Anzeige der verschiedenen Anschlüsse am Computer 14
Abbildung 16: M5Stack-Verbindung wird erfolgreich angezeigt. 14
Abbildung 17: Erklärung zur UIFlow Oberfläche (anhand eines PIR Beispiel). 15
Abbildung 18: Die offizielle MIT App Inventor Webseite 16
Abbildung 19: Die Darstellung, wenn ein neues App Projekt gestartet wird 16
Abbildung 20: Die Designer Ansicht der MIT App Inventor Plattform 17
Abbildung 21: Die Block Ansicht der MIT App Inventor Plattform 17
Abbildung 22: Anzeige der verschiedenen Blöcke in der Designer Ansicht 18
Abbildung 23: Control Block Ansicht 19
Abbildung 24: Variablen Ansicht 20
Abbildung 25: Ein Beispiel verschiedener Variablen 20
Abbildung 26: Layout Ansicht 21
Abbildung 27: Connectivity Ansicht 21
Abbildung 28: Social Ansicht, in der auch Texting vorhanden ist 22
Abbildung 29: Ansicht vom erstellten HorizontalArrangement 23
Abbildung 30: Media Ansicht 24
Abbildung 31: Fenster, welches geöffnet wird, um ein Bild hochzuladen 24
Abbildung 32: QR Code für die MIT AI2 Companion App 25
Abbildung 33: Auf AI Companion klicken, um die Applikation zu erstellen 25
Abbildung 34: Das neue Fenster, dass den QR Code anzeigt, um diesen mit der MIT AI2 Companion App einzuscannen 26
Abbildung 35: M5Stack mit den dazugehörigen Units für das Krisenmanagmentsystem 27
Abbildung 36: Ansicht aller Komponenten unter "Palette" 49
Abbildung 37: Benötigte Komponenten, um die mobile Applikation zu erstellen 50
Abbildung 38: Einstellung von HorizontalArrangement1 51
Abbildung 39: Die serverURL Komponente muss unter HorizontalArrangement1 stehen 51
Abbildung 40: Die Eigenschaften von der serverURL Komponente 52
Abbildung 41: Die erstellte IP-Adresse vom M5Stack wird unter dem Text „Network Config“ angezeigt. 53

Abbildung 42: Die Eigenschaften von der goButton Komponente 53
Abbildung 43: Die Eigenschaften von der liveCheckBox 54
Abbildung 44: Das hochgeladene Bild namens "lineImage" 54
Abbildung 45: Die Eigenschaften der Komponente lineImage.....55
Abbildung 46: Die Eigenschaften der bioDataList Komponente 55
Abbildung 47: Die Eigenschaften der ProjectName Komponente.....56
Abbildung 48: Die Eigenschaften der msgNotifier Komponente 56
Abbildung 49: Die Eigenschaften der webComponent Komponente 57
Abbildung 50: Die Eigenschaften der timerClock Komponente 57



erden erstellt,
.....58
Abbildung 52: Die Methode, wenn der Go! Button gedrückt wird 58
Abbildung 54: Innerhalb der Methode wird der App angewiesen, wie sie während der Laufzeit auf Fehler reagieren soll 59
Abbildung 55: die readData Methode, welche Daten vom M5Stack erhält 59
Abbildung 56: Die webComponent Methode, welche die erhaltenen Daten in eine Liste speichert 60
Abbildung 57: Ansicht und Erklärung von der mobilen Applikation.....61