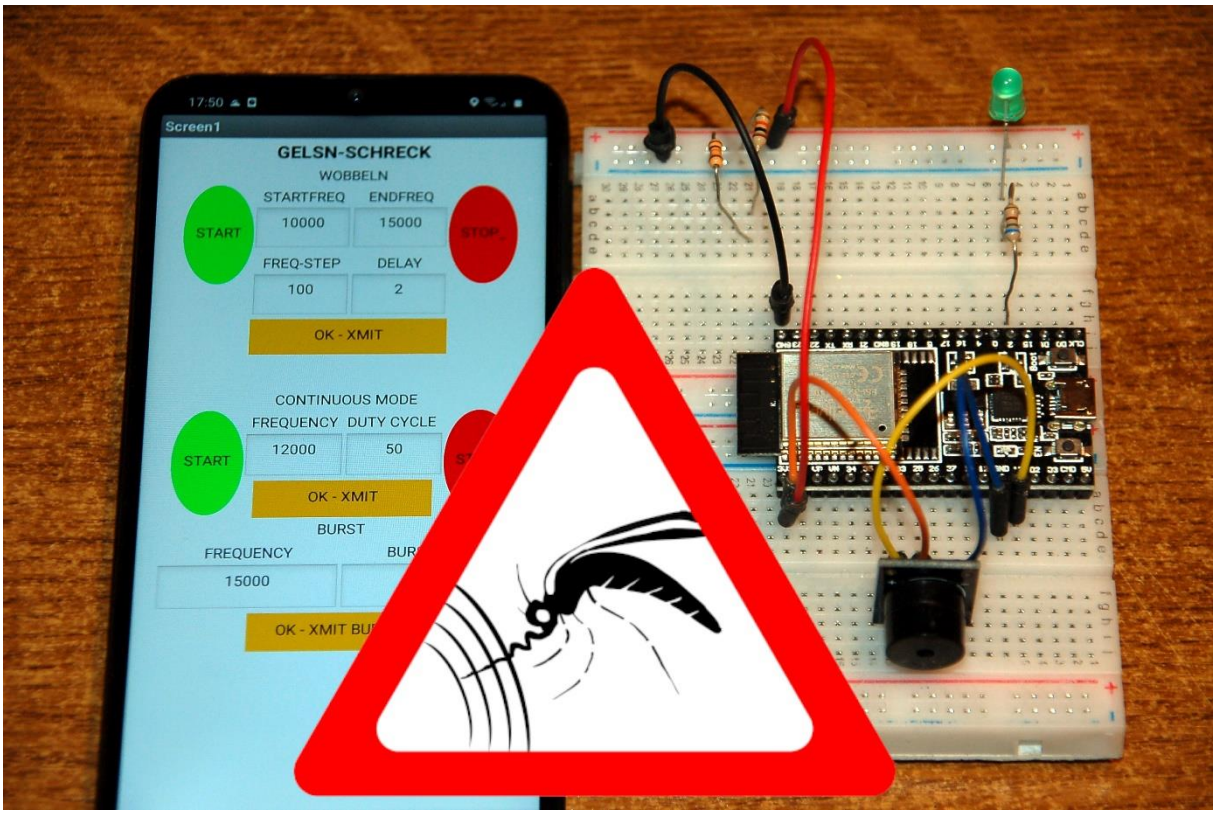

Abbildung 1: Mückenschrecker

Unser ARD – nein nicht Das Erste – sondern unser **A**nimal **R**epellent **D**evice hat bisher die Befehle über die PC-Tastatur bekommen. Das könnte natürlich so bleiben, aber wer möchte sich denn gerne mitsamt PC neben sein Auto setzen, um die AMF (Anti-Marder Frequenz) einzujustieren. Deshalb und weil der ESP32 sonst hoffnungslos unterbeschäftigt wäre, wird jetzt seine Funkschnittstelle in Betrieb genommen. Herzlich willkommen zum Blog mit dem Thema

Die Stech-Mücken-Scheuche – Handy-Applikation

Weil die Bezeichnung so elend lang ist, bleiben wir doch lieber bei SMS-HApp. Mit dem Programm gelsnschreck.py haben wir in der letzten Folge unserem ESP32 beigebracht, wie er Mücken, Marder, Mäuse und anderes Getier dazu bringen kann, das Weite zu suchen. Im Programm gibt es zwei Sektionen, die die Tür zum Funkverkehr öffnen. Vermutlich haben Sie diese Tür schon einmal geöffnet und mit packetsender.exe ein paar Befehle übermittelt. Mit dem PC ist das recht umständlich. Schneller und unkomplizierter geht es mit der App, die wir heute mit dem MIT-AppInventor2 erstellen wollen. Die folgende Liste sagt Ihnen, was Sie dazu brauchen.

Für das Handy

AI2-Companion aus dem Google Play Store.

Für die Handy-App

http://ai2.appinventor.mit.edu
https://ullisroboterseite.de/android-AI2-UDP/UrsAI2UDP.zip
App-Inventor installieren und benutzen – Detaillierte Anleitung
Die fertige App mit der Erweiterung der IP-Eingabe
Die Datei gelsnschrecker.aia, welche den Designer und die Blockdefinitionen enthält

Auf dem ESP32

gelsnschreck.py
Link zur Hard- und Softwarebeschreibung

Die Handy-App

Wir erstellen die App mit Hilfe des Tools AppInvertor2, das unter der MIT-Licence als freie Software über einen Browser (z.B. Firefox) genutzt werden kann. Das bedeutet, dass die Anwendung nicht auf dem PC installiert werden muss, wenn eine Internetverbindung zur Verfügung steht. Wie man damit umgeht, habe ich hier sehr detailliert beschrieben, deshalb gehe ich jetzt nicht näher darauf ein. Auch die Verwendung der UDP-Erweiterung für dieses Tool von Ullis Roboterseite wird dort genau erläutert.

Unser Ziel ist es, unsere bisherigen Befehlsakronyme bestimmten Aktionen auf dem Handy-Bildschirm zuzuordnen und dann an den ESP32 zu senden. Dazu gibt es Eingabefelder und Start- Stop-Buttons. Der Bildschirm ist, wie das Programm gelsnschreck.py, in die drei Befehlsgruppen Wobbeln, Dauerton und Frequenz-Bursts gegliedert. Zahleneingaben werden mit Tippen auf die XMIT-Flächen gesendet, die Buttons lösen die entsprechende Aktion unmittelbar aus. Und so sieht die Bedienoberfläche aus.

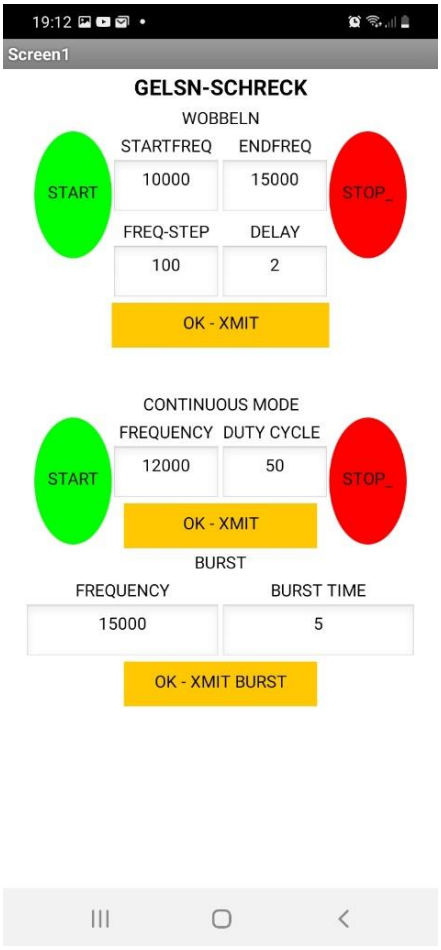

Abbildung 2: Screenshot_ der Oberfläche

Die Projektdatei, gelsnschrecker.aia steht zum Download bereit. Sie können das Projekt über das Menü **My projects – Import project from my Computer** direkt in AI2 importieren.

Denken Sie aber bitte daran, die Netzwerkadresse an Ihre eigene Umgebung anzupassen.

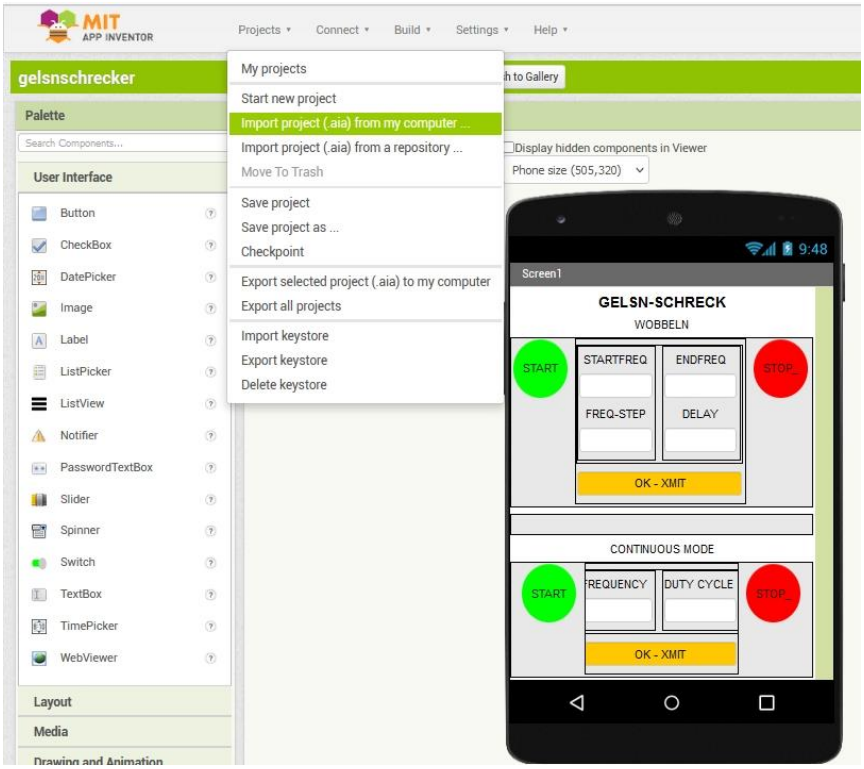

Abbildung 3: AI2

Mit dem AI2 Apps zu erzeugen ist so einfach wie Bausteine stapeln aber, wie diese, manchmal etwas eckig. Das hat sich auch bei diesem Projekt wieder gezeigt. Damit eine Eingabe in ein Textfeld wie STARTFREQ übernommen wird, muss man den Fokus von diesem Feld wegnehmen. Das geht aber nur, indem man einem anderen Textfeld den Fokus zuordnet, indem man es antippt. Eigentlich sollte das Verschieben des Fokus, entsprechend dem Manual, auch mit Buttons funktionieren, tut es aber nicht. Es ist auch nicht möglich, den Fokus einem Textfeld zuzuordnen, das nur gelesen aber nicht beschrieben werden kann. Ein zweites Handicap ist die feste Screengröße, die für neuere Handys zu kurz ist. Trotz dieser Schwachstellen funktioniert die App sehr gut, und wir beginnen auch gleich mit dem trickreichen Aufbau der Oberfläche.

Wir brauchen folgende Elemente.

Bereich Layout:
Horizontal Arrangement
Vertical Arrangement

Bereich User interface:
Button
Label
Textbox

Bereich Extension:
UDPXmitter

Bereiche im Fenster werden durch sogenannte Arrangements vergeben. Horizontal Arrangements erlauben eine Gruppierung von Elementen nebeneinander, Vertical Arrangements ordnen Elemente übereinander an. Der Screen ist im Prinzip ein Vertical Arrangement. Wir starten mit einem Label, dessen Texteigenschaft den Titel der App angibt. Wenn Sie mit Gelsn-Schreck nichts anfangen können, taufen Sie Ihre App doch Mückenschreck. Im niederbayrischen und österreichischen Sprachgebrauch heißen die Mücken Gelsen (oder Gelsn gesprochen "Göisn"), damit können Sie sich jetzt auch den Namen des MicroPython-Programms für den ESP32 erklären. Wenn Sie sich am Rande dieses Blogs für die näheren Umstände einer besonderen Gelsenjagd der anderen Art interessieren sollten, dann kann ich Ihnen den Text zu einem Lied von Ludwig Hirsch ans Herz legen. Ich warne aber von vornherein vor dessen schwarzem Wiener Humor! Unsere Lösung des Problems fällt bei weitem friedlicher und ungefährlicher aus.

Zurück zum AI2. Nach dem Titel der App folgt die Überschrift der ersten Funktionseinheit, WOBBELN, ebenfalls als Label.

Jetzt ist unsere Verpackungslogik gefragt. Die Frage lautet, wie kann ich Schachteln in Schachteln, in Schachteln von Schachteln in einer Schachtel … verpacken? – OK lassen Sie uns das grafisch angehen.

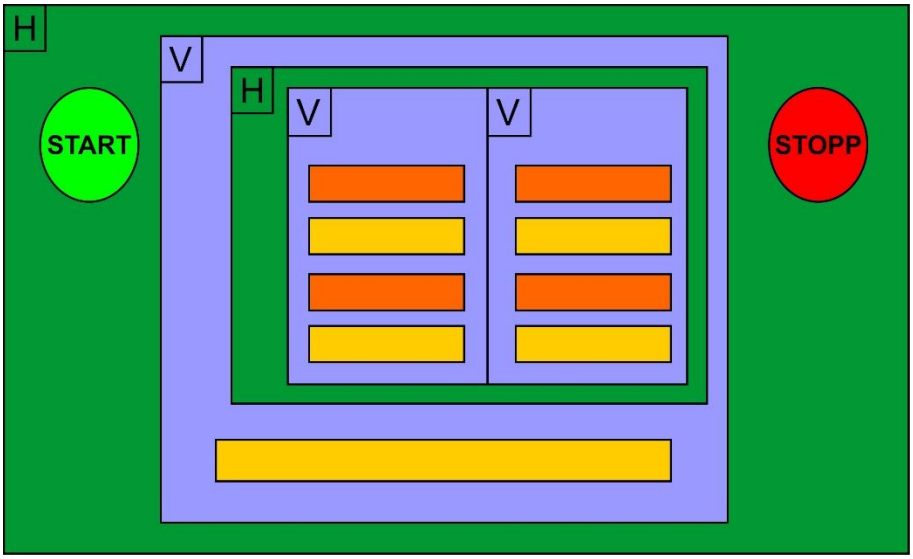

Abbildung 4: Schachtel-Logik

Ohne die Container, die Arrangements, würden alle Elemente untereinander angeordnet. Das verändern wir mit dem großen Horizontal Arrangement, in das wir einen Button, dann ein Vertical Arrangement und schließlich einen zweiten Button packen. Damit unsere Eingabezeilen oberhalb des XMIT-Feldes erscheinen brauchen wir ein weiteres Vertical Arrangement. Unsere Labels und Eingabezeilen

sollen untereinander und in zwei Blöcken nebeneinander erscheinen. Die Grafik sagt uns alles Nötige, das für diese Anordnung gebraucht wird.

Im **Components**-Fenster werden alle Elemente hierarchisch aufgeführt. Sie sehen, dass einige davon eigene Namen tragen wie Start_Wobbeln oder Startfrequenz. Das erreichen Sie mit dem Button **Rename**.

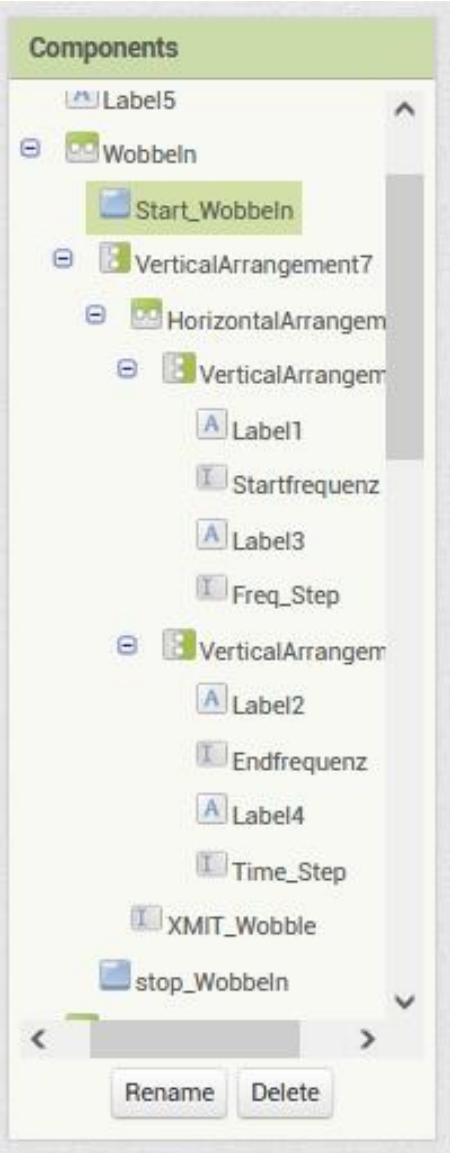

Abbildung 5: Components

Im Fenster Eigenschaften oder Properties wird für jedes Element das Erscheinungsbild festgelegt. Ich denke, die Namen sprechen für sich.

Abbildung 6: Properties

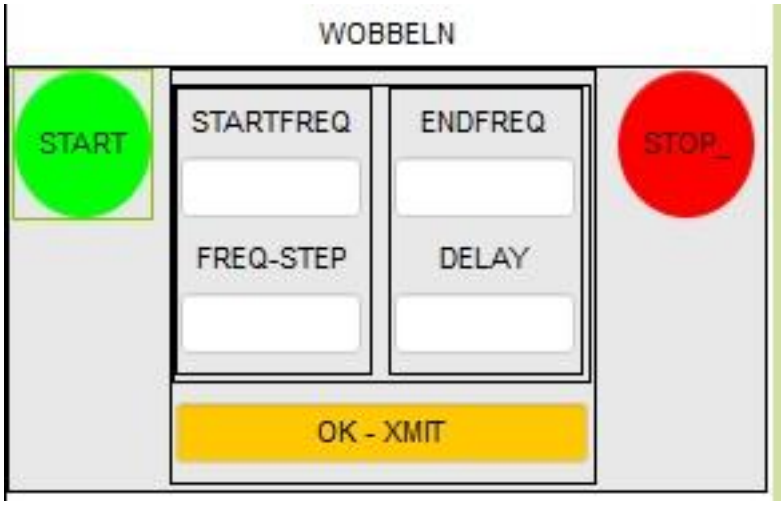

Abbildung 7: Bereich Wobbeln

Wenn Sie bereits den AI2-Companion auf Ihrem Handy installiert haben und über ein lokales Funknetz verfügen, können Sie sich jetzt damit verbinden, um zu sehen, wie der Entwurf auf dem Smartphone wirkt. Die Installation und Bedienung habe ich, wie schon erwähnt, hier beschrieben.

Jetzt bauen wir die Verbindung auf und lassen uns dafür einen QR-Code anzeigen.

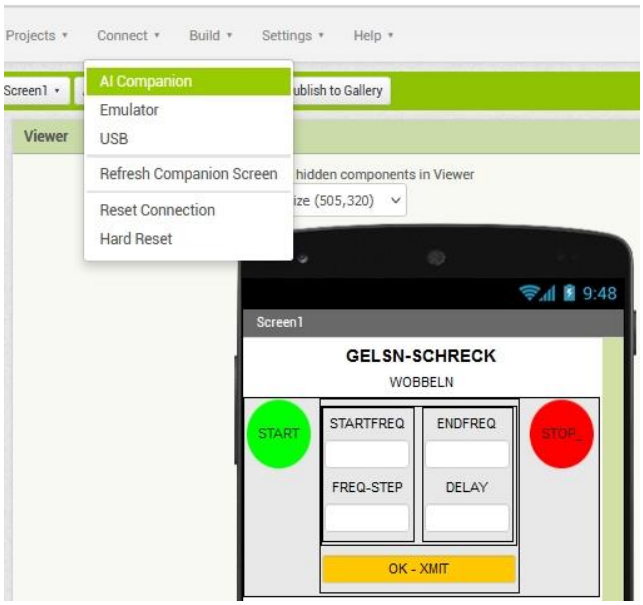

Abbildung 8: Mit dem AI2-Companion verbinden

Abbildung 9: QR-Code für die Verbindung

Auf dem Handy starten wir den AI2 Companion, +tippen auf Scan QR code und halten das Handy gegen den PC-Bildschirm. Nach ein paar Sekunden erscheint unser Entwurf im Handy-Display. Alles, was ab nun auf dem PC geändert wird, erscheint mit kurzer Verzögerung auch auf dem Handy. Erfolgt eine Zeit lang keine Interaktion, wird die Verbindung gecancelt und muss neu aufgebaut werden.

Die Strukturen für die beiden anderen Bereiche, Dauerton und Burstmodus, werden in ähnlicher Weise erstellt. Man muss allerdings zwischenzeitlich die Höhe des Wobble-Arrangements auf 10% setzen, damit man am dritten Teil arbeiten kann. Alternativ kann man im Viewer auf "Tablet size" umstellen.

So sieht die Komponentendarstellung der beiden anderen Bereiche aus. Die Eigenschaften können Sie nach Ihren Wünschen einstellen, das beeinflusst die Funktion der Elemente (meistens) nicht.

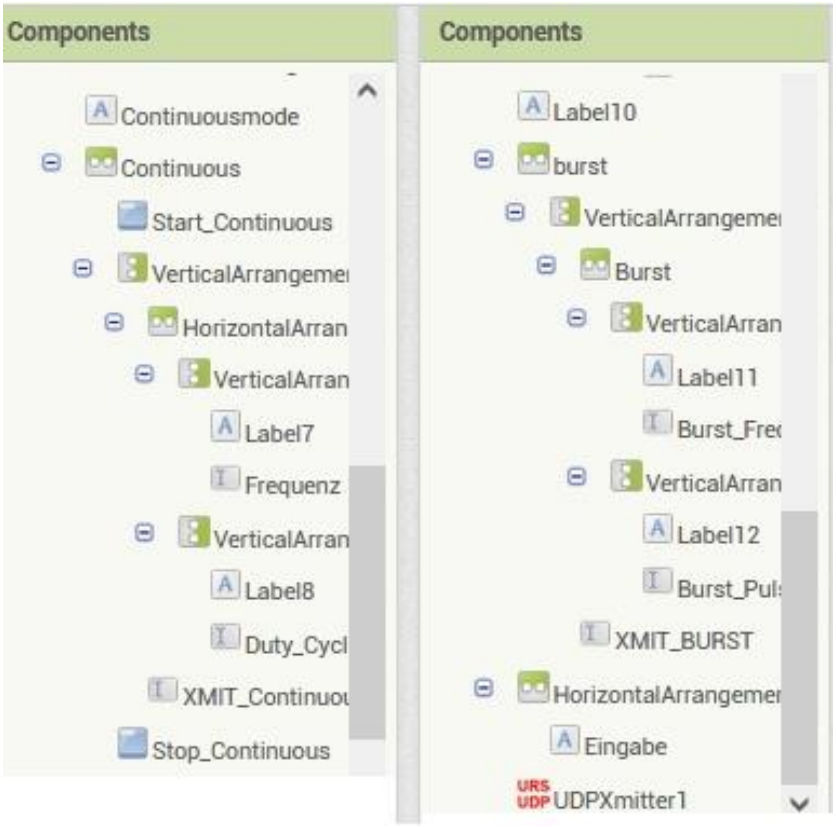

Abbildung 10: Components 2

In der rechten Spalte sehen Sie ganz unten die, von der Erweiterung von Ullis Roboterseite, eingefügte UDP-Funktionalität, die einen UDP-Client zur Verfügung stellt.

Jetzt werden Bausteine gestapelt und damit die App zum Leben erweckt. Beginnen wir wie bei jedem anderen Programm mit der Deklaration einiger Variablen. Die Bausteine werden aus dem Ordner Variables gezogen, Zahlen kommen aus Math und das Textelement aus Text.

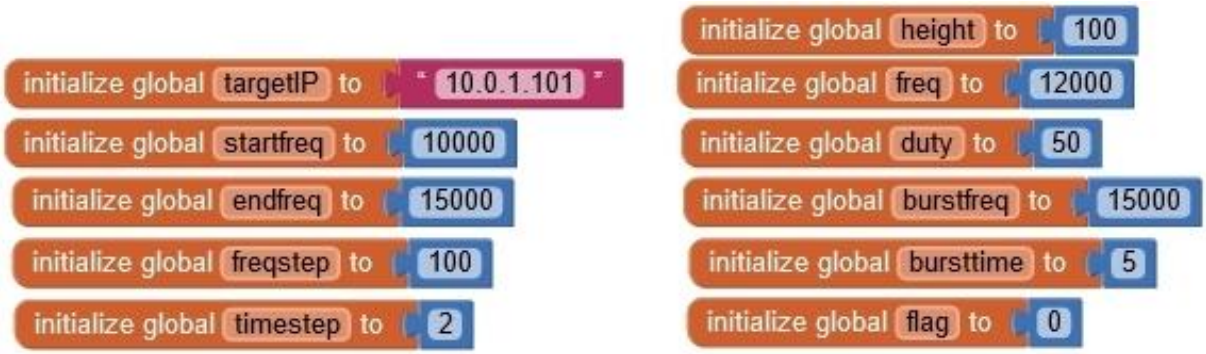

Abbildung 11: Variablen festlegen

Beim Aufbau der Oberfläche sollen die Felder mit den Variablenwerten belegt werden. Wir holen uns aus dem Ordner **Screen1** eine **when Screen1.initialize**-Klammer und füllen diese mit den Bausteinen zum Belegen der Texteigenschaften aus dem Ordner des jeweiligen Elements. Die folgende Abbildung zeigt das Setzen der Ziel-IP-Adresse unseres ESP32.

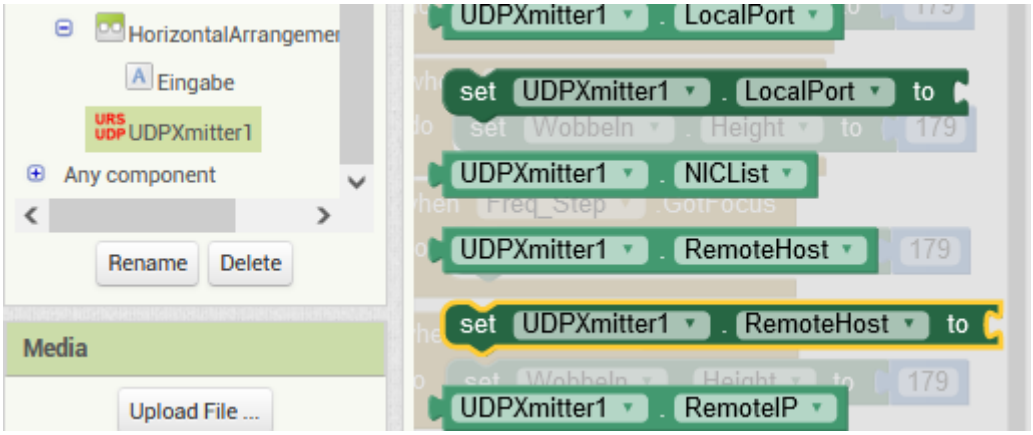

Abbildung 12: UDP-Zieladresse setzen

Abbildung 13: Felder füllen

Wenden wir uns dem Bereich Wobbeln zu. Beim Tippen auf das Textfeld OK-XMIT passiert Dreierlei. Das Feld bekommt den Fokus während ihn eines der Eingabefelder verliert. Das hilft uns, festzustellen, welches Feld ihn hatte. Wir kennzeichnen das mit einer Nummer und merken uns die Eingabe in der entsprechenden Variablen. Wurde dagegen einer der Buttons getippt, senden wir sofort die entsprechende Nachricht an den ESP32.

Abbildung 14: Aktionen im Bereich Wobbeln

Wer den Fokus bekommt, ist als Nächster dran und das ist das Textfeld **XMIT_Wobble**. Wenn es den Fokus bekommt, schaut die Codesequenz **when XMIT_Wobble.GotFocus** über die globale Variable **flag** nach, wer ihn abgegeben hat und erledigt dann den Auftrag. Das Befehlsakronym und der entsprechende Variableninhalt werden gesendet und danach erhält flag den Wert 0. Eingegebene Werte werden im Label **Eingabe** zur Kontrolle angezeigt.

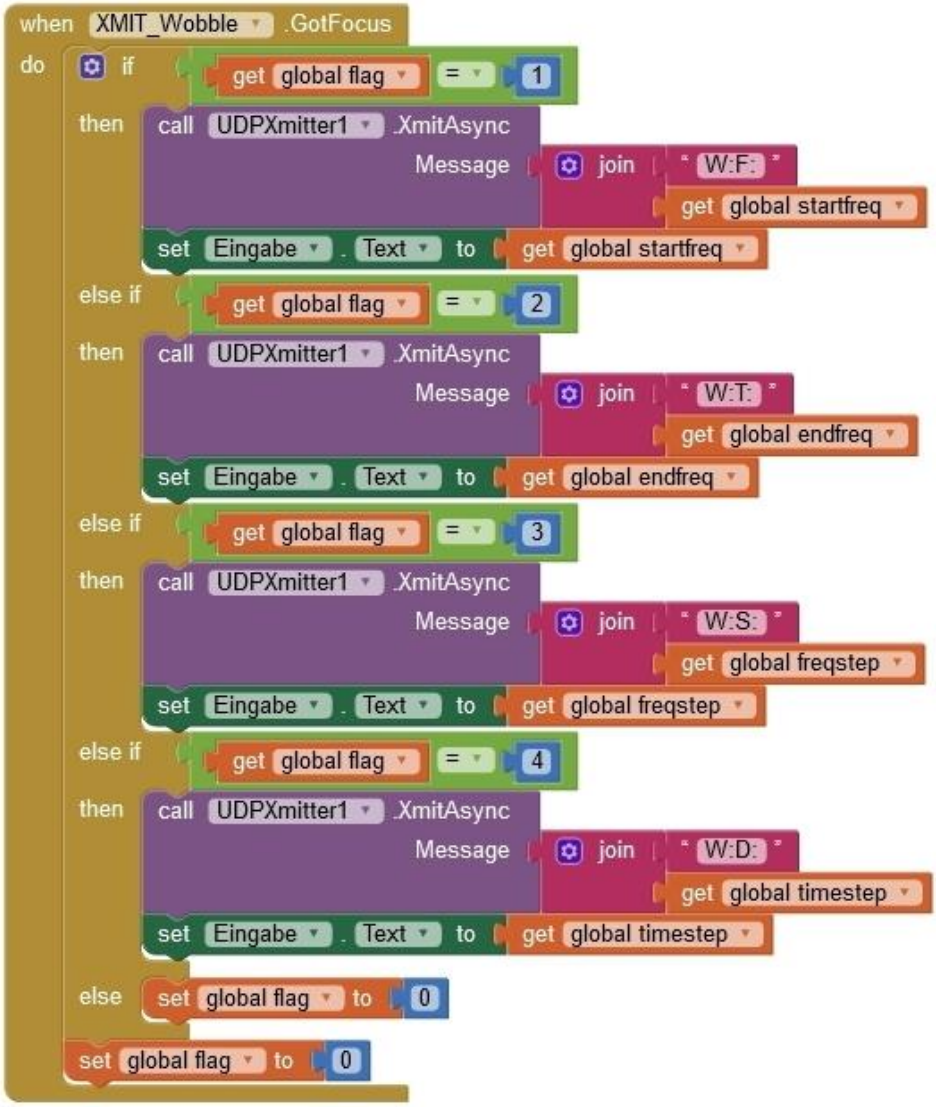

Abbildung 15: Wobbeln XMIT wurde angetippt

Die Abteilung Dauerfrequenz arbeitet nach demselben Schema.

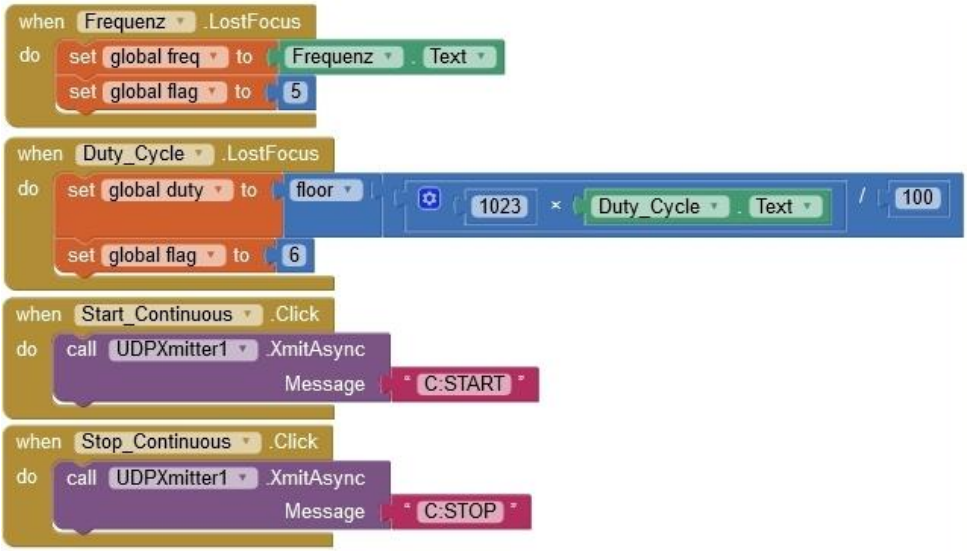

Abbildung 16: Dauerton-Steuerblock 1

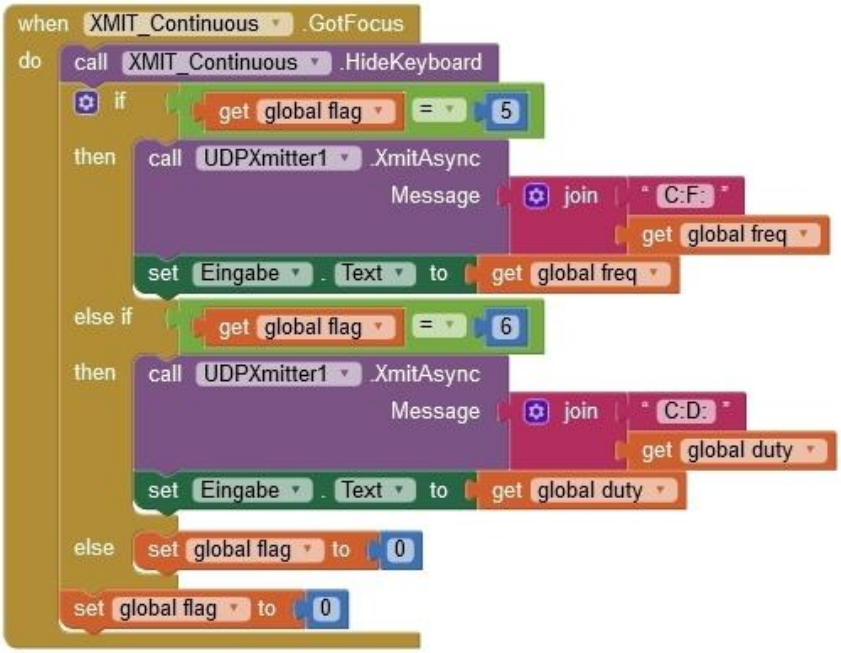

Abbildung 17: Dauerton-Steuerblock 2

Die Eingabe von Werten im Burstbereich ist nur mit einem Trick machbar. Beim Antippen eines Eingabefeldes verschwindet es nämlich hinter der Tastatur. Wird nun eine Zahl blind eingegeben, so ist dennoch das Feld hernach leer, die Zahl hat sich ins Nirvana verabschiedet. Jetzt könnte man einen zweiten Screen hernehmen oder, wie ich das gelöst habe, den oberen Teil soweit zusammenschieben, dass der Burstbereich genug Platz hat. Genau das passiert, sobald eines der Textfelder in diesem Bereich den Fokus erhält, geht das horizontal Arrangement Wobbeln auf die Höhe 10%. Die Verarbeitung der Eingaben erfolgt wieder genauso wie bei den beiden anderen Bereichen.

Abbildung 18: Burst-Steuerung 1

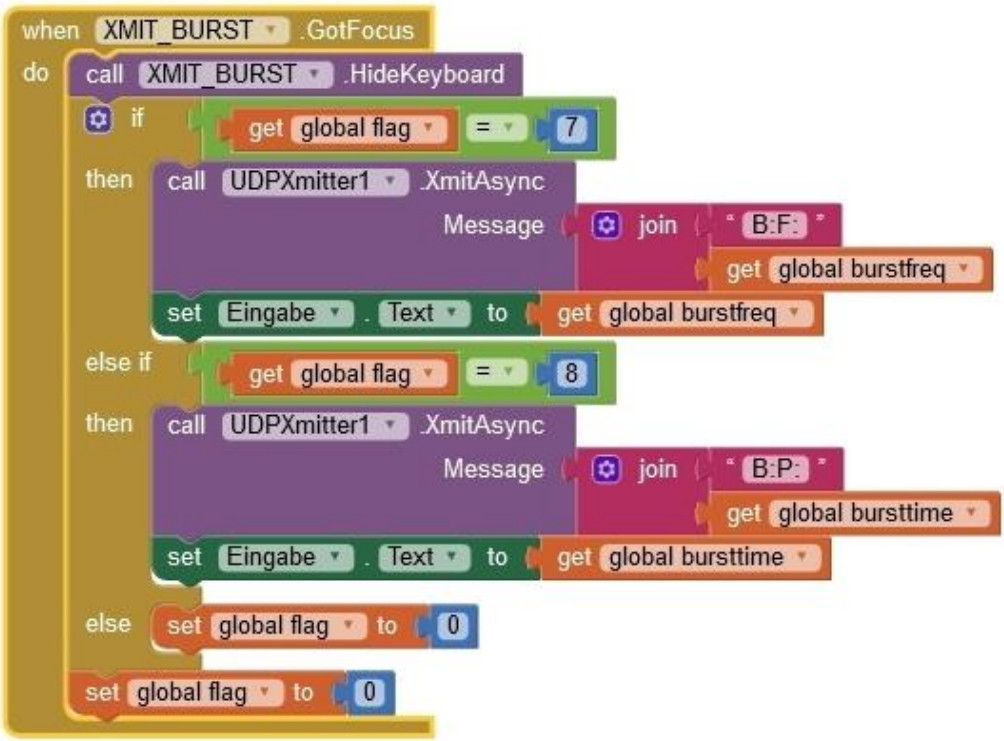

Abbildung 19: Burst-Steuerung 2

Natürlich muss der Wobbelbereich wieder vergrößert werden, wenn dort Eingaben erfolgen sollen. Weil es keine Möglichkeit gibt, den Wert für die Höhe wieder auf **automatisch** zu setzen, wie beim Entwurf, muss man die Höhe als Zahlenwert angeben.

Abbildung 20: Wobbeln groß schalten

Wenn wir keine Warnungen oder Fehlermeldungen haben, gehen wir ans Compilieren der App.

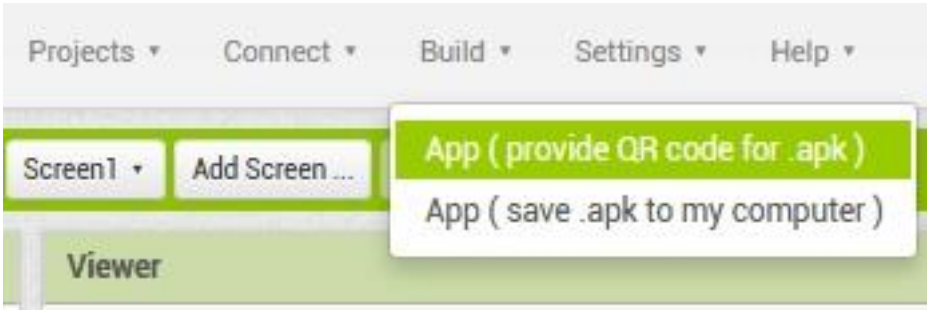

Abbildung 21: Build App and install

Wir können entweder einen QR-Code anfordern, um die App nach Fertigstellung direkt aufs Handy zu laden, oder die apk-Datei auf dem PC speichern, um sie, etwa durch Bluetooth, später zu übertragen oder zu teilen. Folgen Sie bitte für das Herunterladen der Anleitung ab Seite 20. Ein Listing in Textform gibt es zu diesem Teil des Projekts nicht, es wird durch die Abbildung der Blöcke ersetzt.

Ich wünsche fröhliches Mücken- und Marderscheuchen.