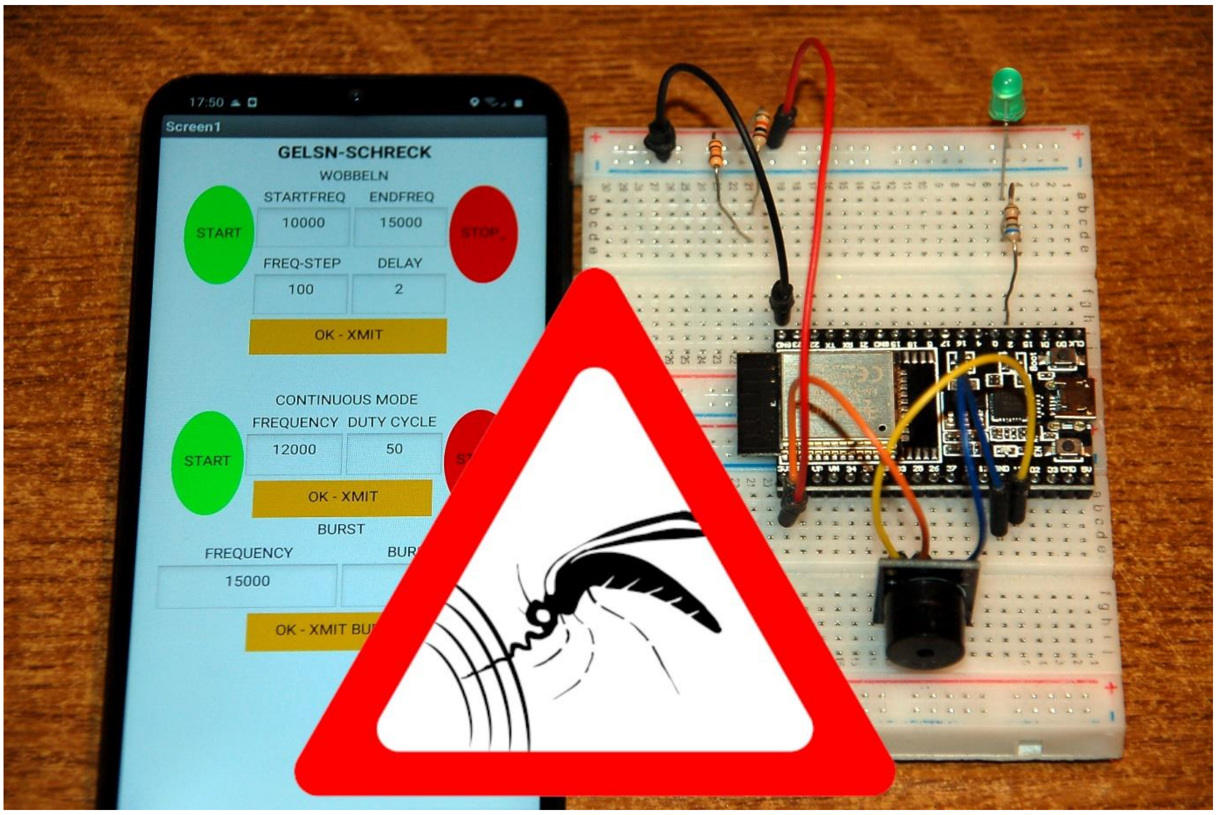


Abbildung 1: Mückenschrecker

Our ARD - no not the first - but our Animal Repellent Device has so far received the commands via the PC keyboard. It could stay that way, of course, but who would like to sit down next to their car with their PC to adjust the AMF (anti-marten frequency). For this reason and because the ESP32 would otherwise be hopelessly underemployed, its radio interface is now being put into operation. Welcome to the blog on the topic

The mosquito repeller - mobile phone application

Because the name is so miserably long, we prefer to stick with SMS-HApp. With the program gelsnschreck.py we taught our ESP32 in the last episode how it can get mosquitoes, martens, mice and other animals to run away. There are two sections in the program that open the door to radio communication. You have probably already opened this door and sent a few commands with packetsender.exe. With the PC this is quite cumbersome. It's faster and less complicated with the app that we want to create today with the MIT-AppInventor2. The following list tells you what you need to do this.

For the cell phone

AI2-Companion aus dem Google Play Store.

For the mobile 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

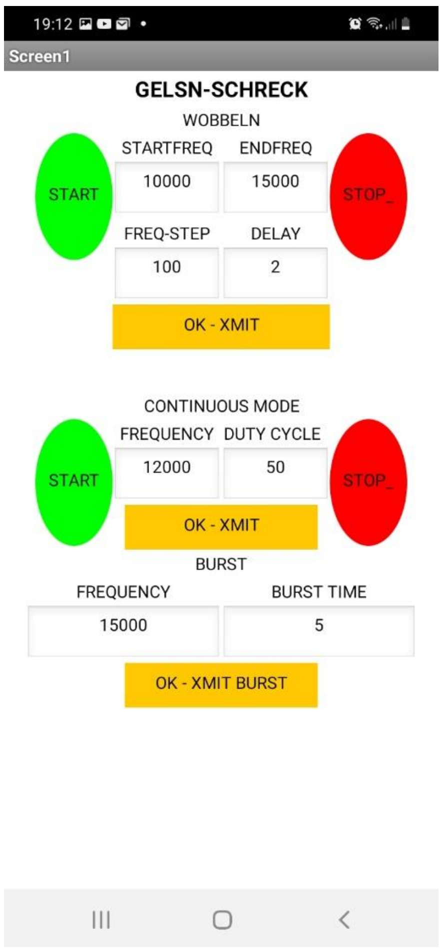
The ESP32

gelsnschreck.py
Link to Hard- und Softwarebeschreibung

The mobile app

We create the app with the help of the AppInvertor2 tool, which can be used under the MIT license as free software via a browser (e.g. Firefox). This means that the application does not need to be installed on the PC if an internet connection is available. How to deal with it I have described in great detail here, so I will not go into it in more detail now. The use of the UDP extension for this tool from Ullis robot site is also explained in detail there.

Our goal is to assign our previous command acronyms to specific actions on the mobile phone screen and then to send them to the ESP32. There are also input fields and start-stop buttons. Like the gelsnschreck.py program, the screen is divided into three command groups: wobbling, continuous tone and frequency bursts. Entering numbers is sent by tapping on the XMIT surfaces, the buttons trigger the corresponding action immediately. And this is what the user interface looks like.



GELSN-SCHRECK

WOBBELN

STARTFREQ	ENDFREQ
10000	15000
FREQ-STEP	DELAY
100	2

CONTINUOUS MODE

FREQUENCY	DUTY CYCLE
12000	50

BURST

FREQUENCY	BURST TIME
15000	5

Abbildung 2: Screenshot_ der Oberfläche

The project file, gelsnschrecker.aia, is available for download. You can import the project directly into AI2 via the menu My projects - Import project from my Computer.

But please remember to adapt the network address to your own environment.

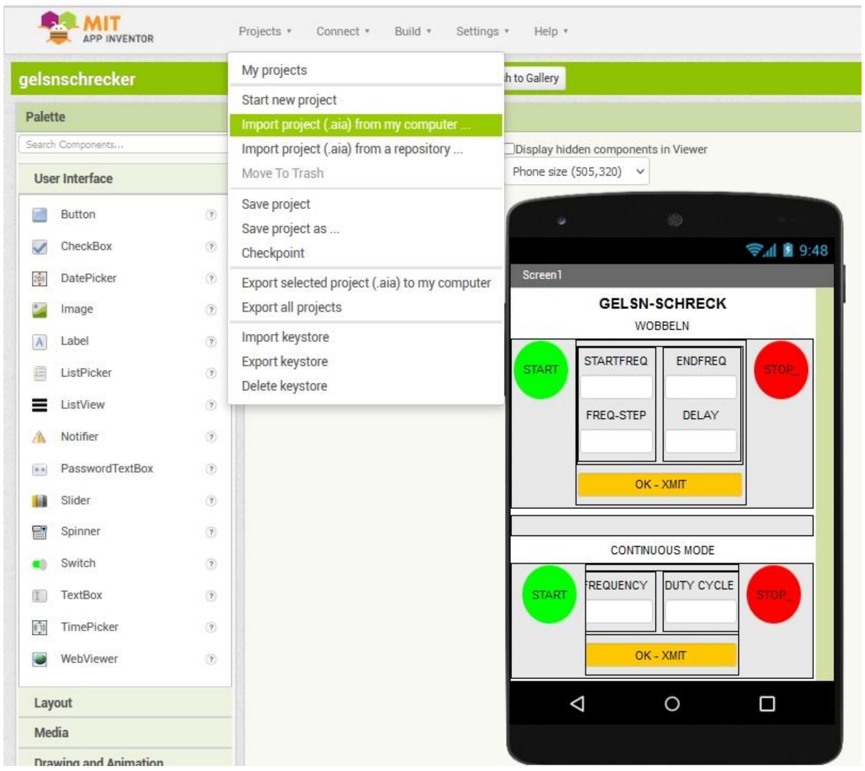


Abbildung 3: AI2

Creating apps with the AI2 is as easy as stacking building blocks but, like these, sometimes a bit angular. This has also been shown again in this project. In order for an entry in a text field such as STARTFREQ to be accepted, the focus must be removed from this field. But this is only possible by assigning the focus to another text field by tapping it. Actually, according to the manual, shifting the focus should also work with buttons, but it doesn't. It is also not possible to assign the focus to a text field that can only be read but not written to. A second handicap is the fixed screen size, which is too short for newer cell phones. Despite these weaknesses, the app works very well, and we'll start with the tricky structure of the interface.

We need the following elements.

Layout area:
Horizontal arrangement
Vertical arrangement

User interface area:
Button
Label
Text box

Extension area:
UDPXmitter

Areas in the window are assigned by so-called arrangements. Horizontal arrangements allow elements to be grouped next to one another, while vertical arrangements arrange elements on top of one another. The screen is basically a vertical arrangement. We start with a label whose text property specifies the title of the app. If you don’t know what to do with Gelsn-Schreck, give your app the name Mosquito Schreck. In Lower Bavarian and Austrian usage, the mosquitoes are called Gelsen (or Gelsn, spoken "Göisn"), so you can now explain the name of the MicroPython program for the ESP32. If, in the margins of this blog, you are interested in the details of a special kind of gelato hunt, then I can recommend the text to a song by Ludwig Hirsch to you. But I warn from the outset against its black Viennese humor! Our solution to the problem is far more peaceful and harmless.

Back to the AI2. The title of the first functional unit, WOBBELN, follows the title of the app, also as a label.

Now our packaging logic is in demand. The question is, how can I pack boxes in boxes, boxes of boxes in a box ...? - OK let's do this graphically.

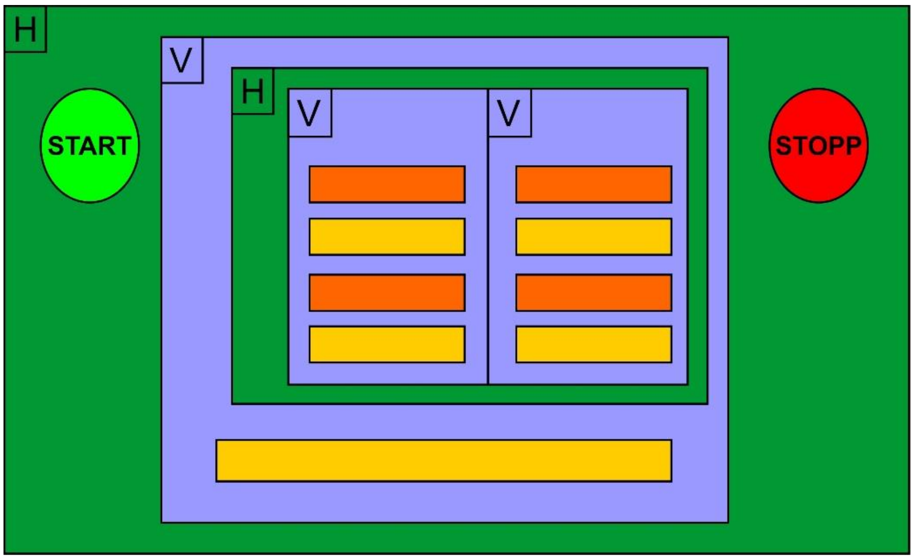
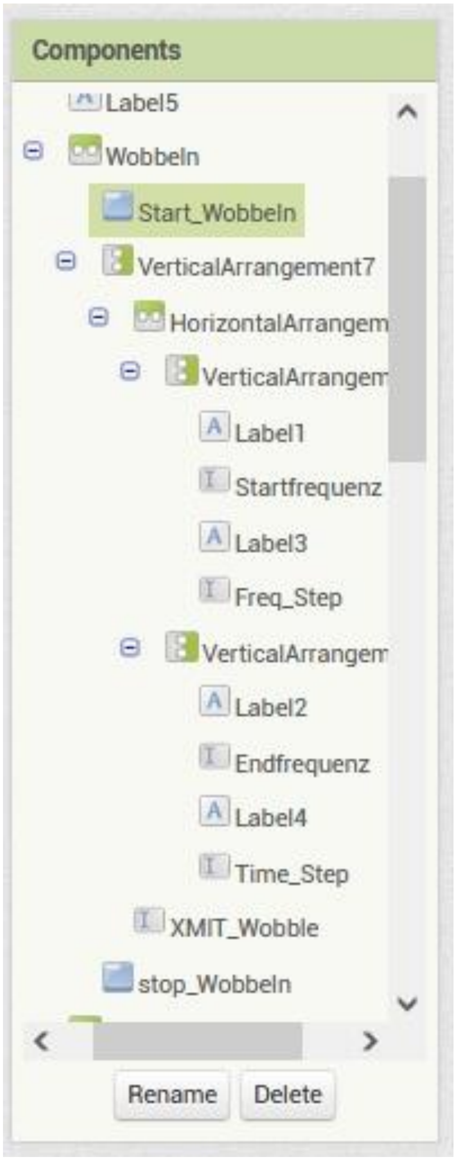


Abbildung 4: Schachtel-Logik

Without the containers, the arrangements, all elements would be arranged one below the other. We change that with the large horizontal arrangement in which we put a button, then a vertical arrangement and finally a second button. So that our input lines appear above the XMIT field, we need another vertical arrangement. Our labels and input lines should appear one below the other and in two blocks next to each other. The graphic tells us everything that is needed for this arrangement.

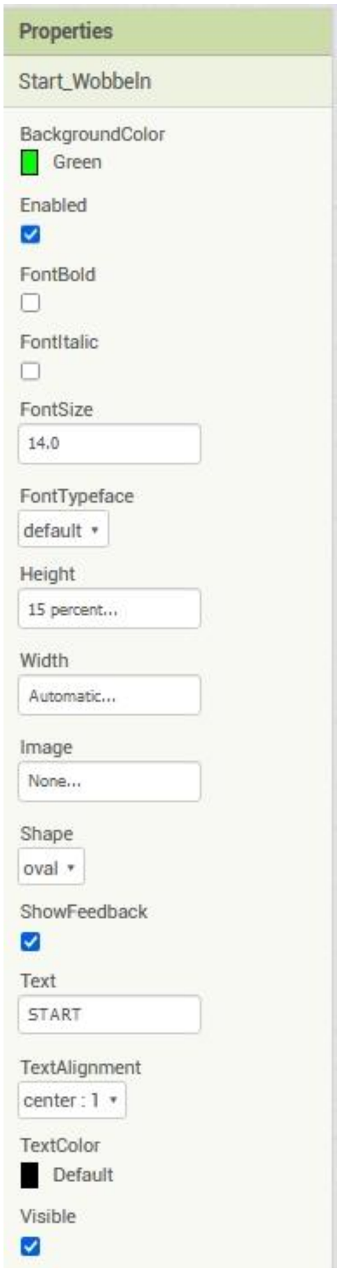
All elements are listed hierarchically in the **Components** window. You can see that some of them have their own names like **Start_Wobbeln** or **Startfrequency**. You can do this with the **Rename** button.



Label5
Wobbeln
Start_Wobbeln
VerticalArrangement7
HorizontalArrangem
VerticalArrangem
Label1
Startfrequenz
Label3
Freq_Step
VerticalArrangem
Label2
Endfrequenz
Label4
Time_Step
XMIT_Wobble
stop_Wobbeln

Abbildung 5: Components

The appearance of each element is defined in the Properties or Properties window. I think the names speak for themselves.



Properties

Start_Wobbeln

BackgroundColor
🟩 Green

Enabled
☑

FontBold
☐

FontItalic
☐

FontSize
14.0

FontTypeface
default

Height
15 percent...

Width
Automatic...

Image
None...

Shape
oval

ShowFeedback
☑

Text
START

TextAlignment
center : 1

TextColor
■ Default

Visible
☑

Abbildung 6: Properties

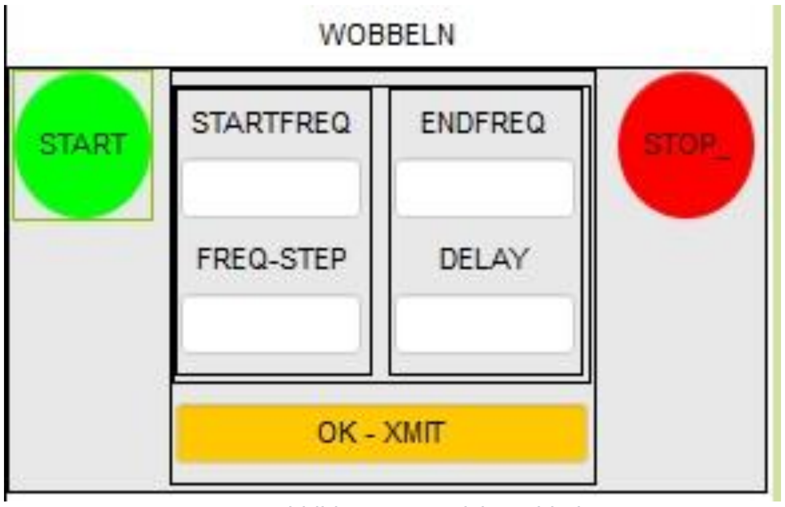


Abbildung 7: Bereich Wobbeln

If you have already installed the AI2 Companion on your mobile phone and have a local wireless network, you can now connect to it to see how the design looks on the smartphone. As already mentioned, I have described the installation and operation here.

Now we set up the connection and display a QR code for it.

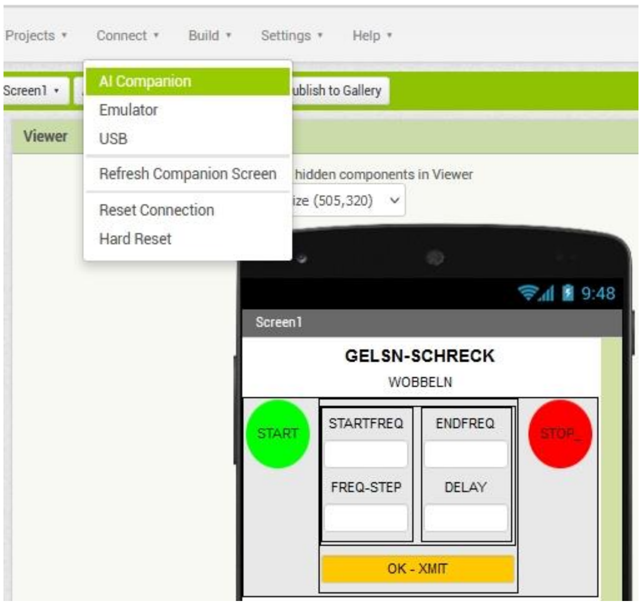
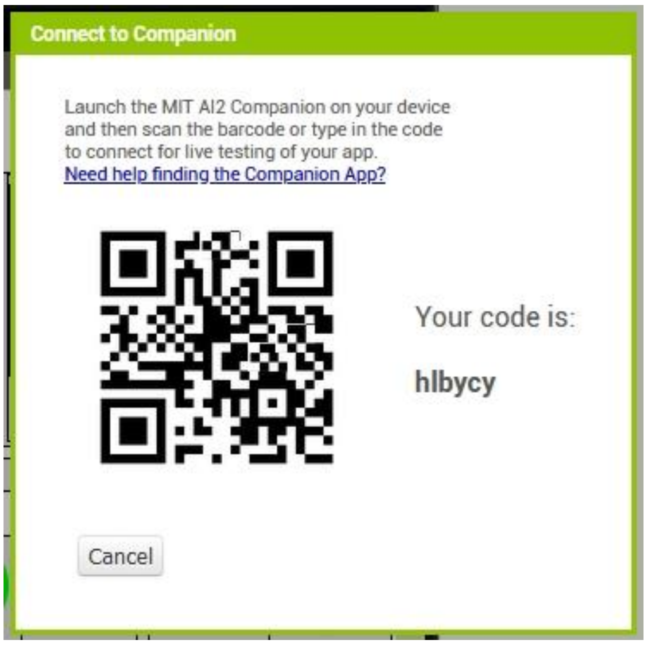


Abbildung 8: Mit dem AI2-Companion verbinden


Connect to Companion

Launch the MIT AI2 Companion on your device and then scan the barcode or type in the code to connect for live testing of your app.
Need help finding the Companion App?

Your code is:

hlbycy

Cancel

Abbildung 9: QR-Code für die Verbindung

We start the AI2 Companion on the mobile phone, + tap Scan QR code and hold the mobile phone against the PC screen. After a few seconds, our draft will appear on the mobile phone display. Everything that is changed on the PC from now on also appears on the mobile phone with a short delay. If there is no interaction for a while, the connection is canceled and must be re-established.

The structures for the other two areas, continuous tone and burst mode, are created in a similar way. In the meantime, however, you have to set the level of the wobble arrangement to 10% so that you can work on the third part. Alternatively, you can switch to "Tablet size" in the viewer.

This is what the component representation of the other two areas looks like. You can set the properties according to your wishes, this (mostly) does not affect the function of the elements.

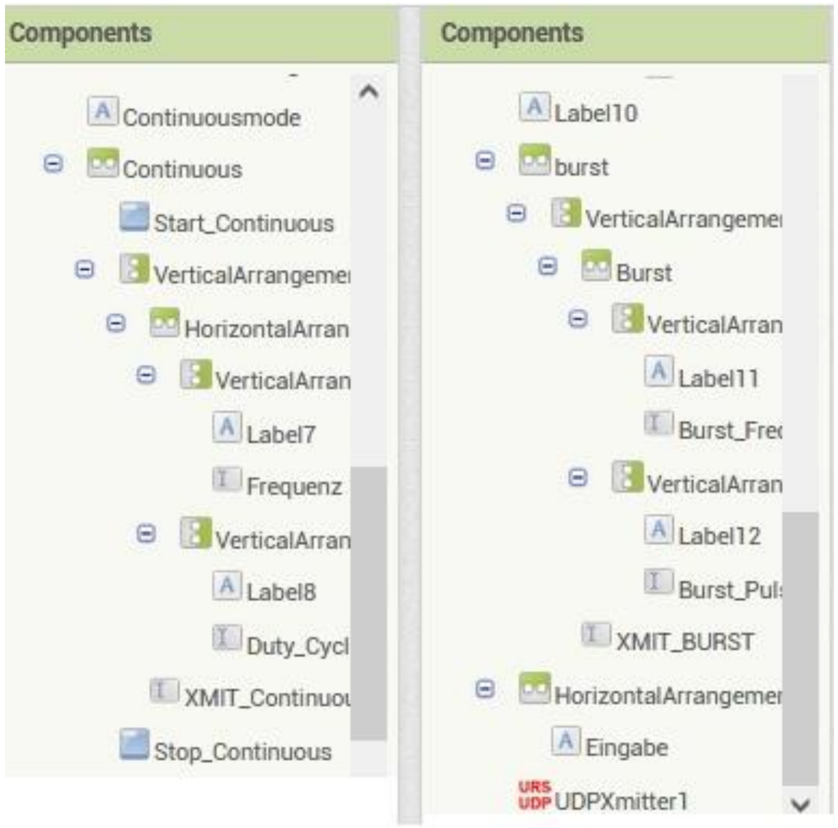
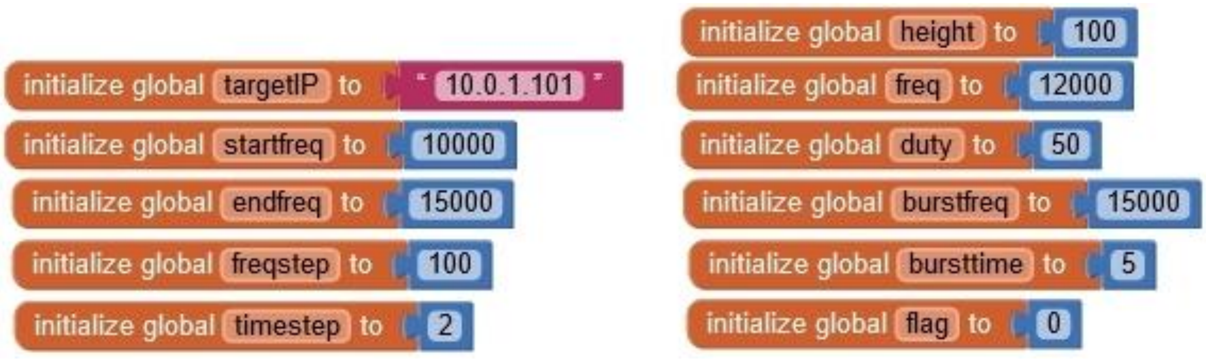


Abbildung 10: Components 2

In the right column at the bottom you can see the UDP functionality added by the Ullis robot page extension, which makes a UDP client available.

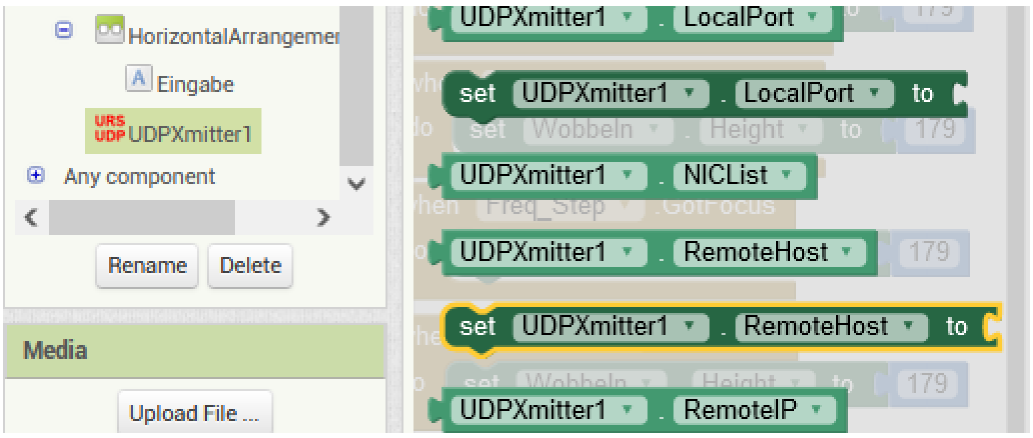
Now the building blocks are stacked and the app is brought to life. As with any other program, let's start by declaring a few variables. The building blocks are drawn from the Variables folder, numbers come from Math and the text element from text.



```
initialize global targetIP to "10.0.1.101"
initialize global startfreq to 10000
initialize global endfreq to 15000
initialize global freqstep to 100
initialize global timestep to 2

initialize global height to 100
initialize global freq to 12000
initialize global duty to 50
initialize global burstfreq to 15000
initialize global bursttime to 5
initialize global flag to 0
```

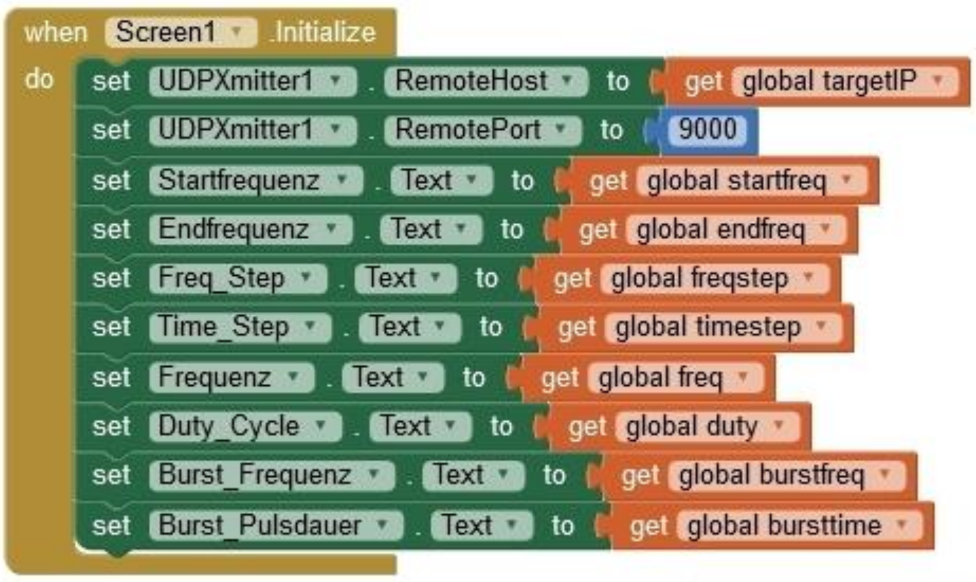
Abbildung 11: Variablen festlegen

When creating the user interface, the fields should be assigned the variable values. We get a when Screen1.initialize bracket from the Screen1 folder and fill this with the blocks for assigning the text properties from the folder of the respective element. The following figure shows the setting of the destination IP address of our ESP32.



```
UDPXmitter1 . LocalPort
set UDPXmitter1 . LocalPort to
UDPXmitter1 . NICList
UDPXmitter1 . RemoteHost
set UDPXmitter1 . RemoteHost to
UDPXmitter1 . RemoteIP
```

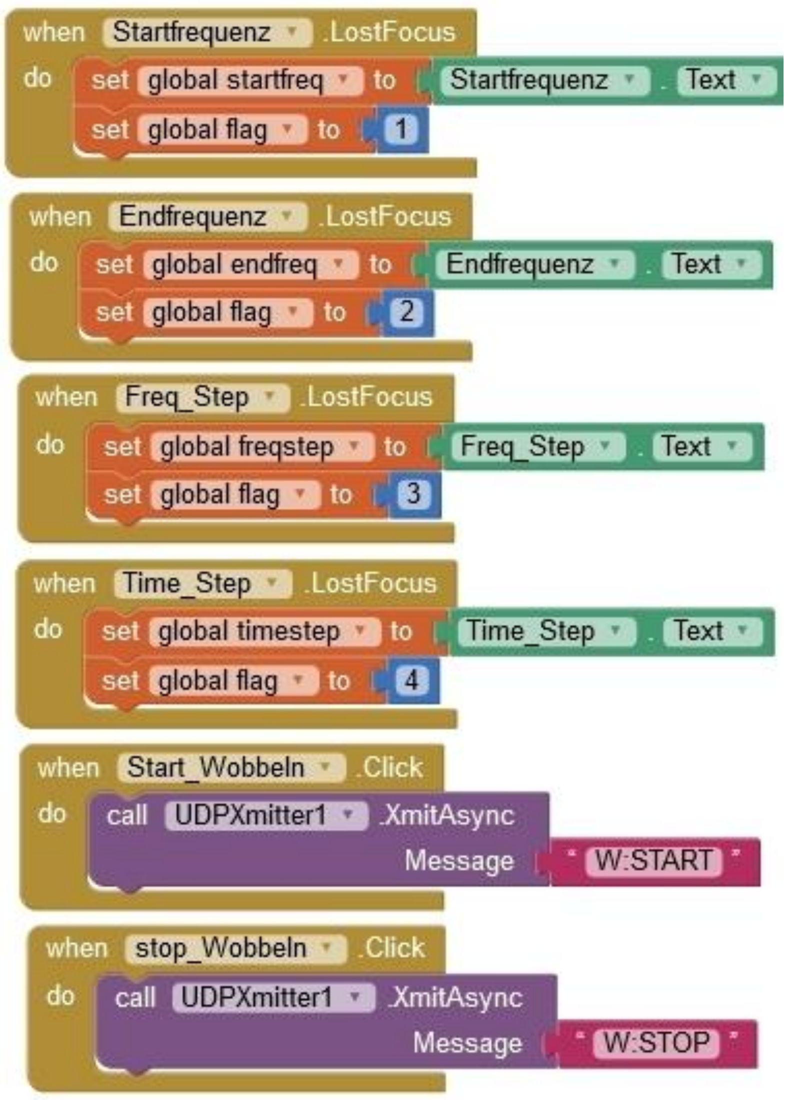
Abbildung 12: UDP-Zieladresse setzen



```
when Screen1 .Initialize
do  set UDPXmitter1 . RemoteHost to get global targetIP
    set UDPXmitter1 . RemotePort to 9000
    set Startfrequenz . Text to get global startfreq
    set Endfrequenz . Text to get global endfreq
    set Freq_Step . Text to get global freqstep
    set Time_Step . Text to get global timestep
    set Frequenz . Text to get global freq
    set Duty_Cycle . Text to get global duty
    set Burst_Frequenz . Text to get global burstfreq
    set Burst_Pulsdauer . Text to get global bursttime
```

Abbildung 13: Felder füllen

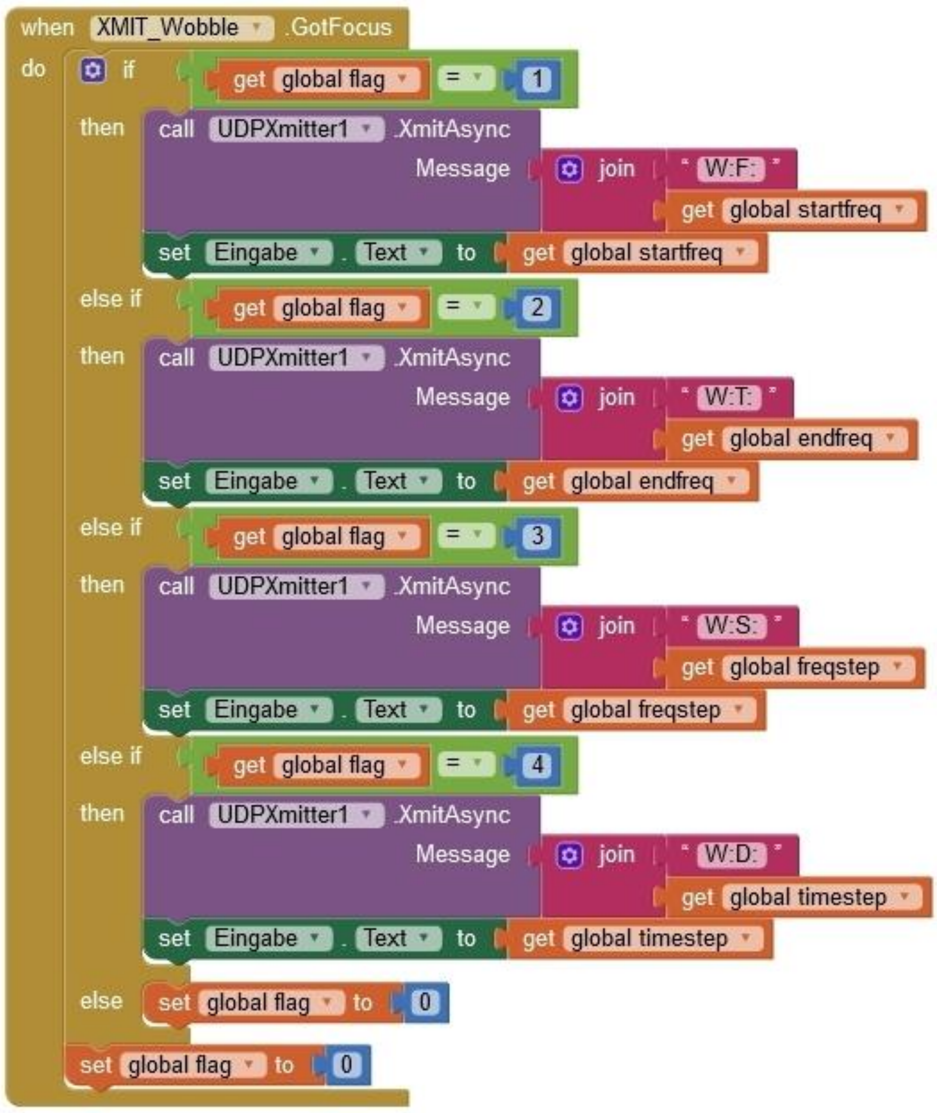
Let's turn to the wobble area. When you tap the OK-XMIT text field, three things happen. The field gets the focus while one of the input fields loses it. This helps us determine which field had it. We mark this with a number and note the entry in the corresponding variable. If, on the other hand, one of the buttons was tapped, we immediately send the corresponding message to the ESP32.



```
when Startfrequenz .LostFocus
do  set global startfreq to Startfrequenz . Text
    set global flag to 1

when Endfrequenz .LostFocus
do  set global endfreq to Endfrequenz . Text
    set global flag to 2

when Freq_Step .LostFocus
do  set global freqstep to Freq_Step . Text
    set global flag to 3

when Time_Step .LostFocus
do  set global timestep to Time_Step . Text
    set global flag to 4

when Start_Wobbeln .Click
do  call UDPXmitter1 .XmitAsync
                     Message " W:START "

when stop_Wobbeln .Click
do  call UDPXmitter1 .XmitAsync
                     Message " W:STOP "
```

Abbildung 14: Aktionen im Bereich Wobbeln

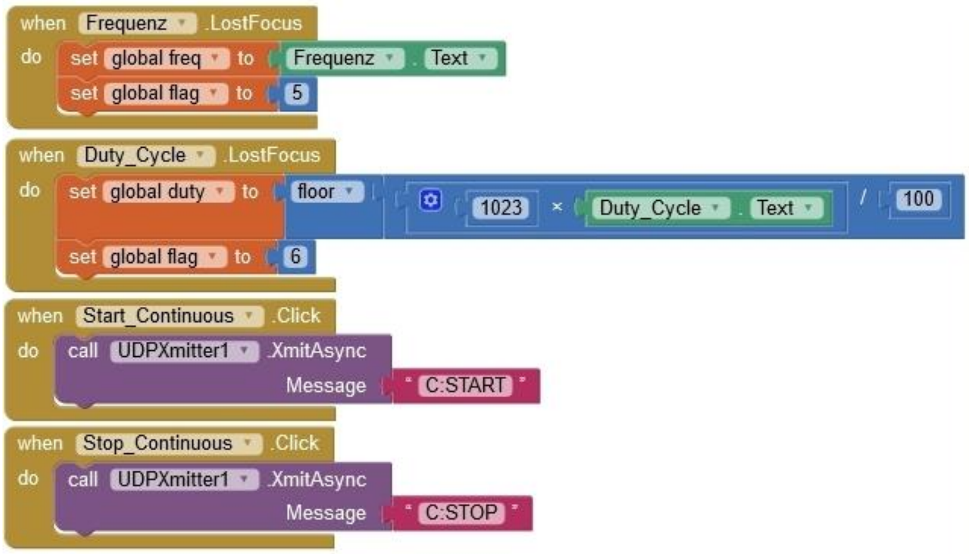
Whoever gets the focus is next and that is the text field XMIT_Wobble. When it gets the focus, the code sequence when XMIT_Wobble.GotFocus uses the global variable flag to see who submitted it and then does the job. The command acronym and the corresponding variable content are sent and then flag receives the value 0. Entered values are displayed in the input label for control.



```
when XMIT_Wobble .GotFocus
do  if   get global flag = 1
    then call UDPXmitter1 .XmitAsync
              Message  join  " W:F: "
                             get global startfreq
         set Eingabe . Text to  get global startfreq
    else if  get global flag = 2
    then call UDPXmitter1 .XmitAsync
              Message  join  " W:T: "
                             get global endfreq
         set Eingabe . Text to  get global endfreq
    else if  get global flag = 3
    then call UDPXmitter1 .XmitAsync
              Message  join  " W:S: "
                             get global freqstep
         set Eingabe . Text to  get global freqstep
    else if  get global flag = 4
    then call UDPXmitter1 .XmitAsync
              Message  join  " W:D: "
                             get global timestep
         set Eingabe . Text to  get global timestep
    else set global flag to 0
    set global flag to 0
```

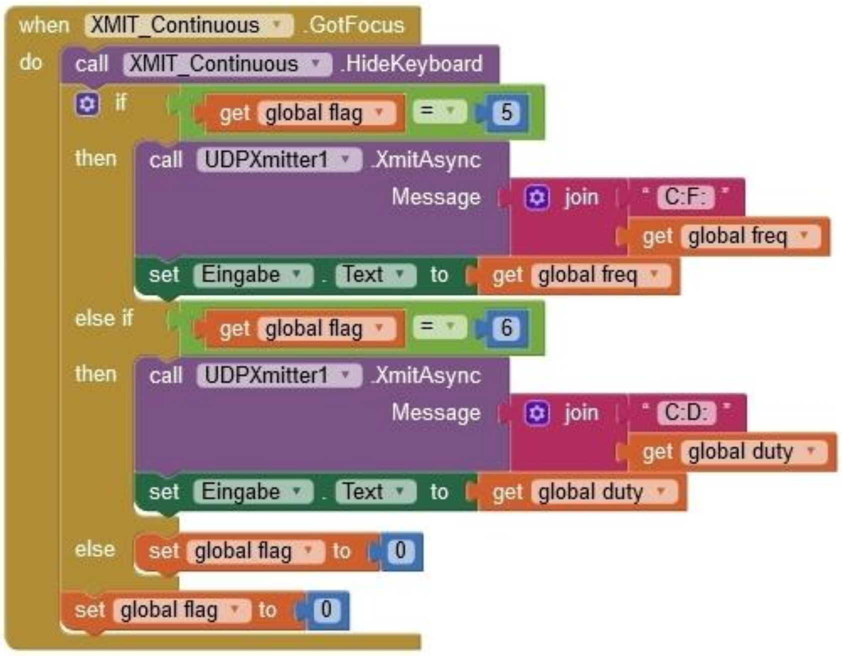
Abbildung 15: Wobbeln XMIT wurde angetippt

The Continuous Frequency department works according to the same scheme.



```
when Frequenz .LostFocus
do  set global freq to  Frequenz . Text
    set global flag to 5

when Duty_Cycle .LostFocus
do  set global duty to  floor  1023 × Duty_Cycle . Text / 100
    set global flag to 6

when Start_Continuous .Click
do  call UDPXmitter1 .XmitAsync
         Message  " C:START "

when Stop_Continuous .Click
do  call UDPXmitter1 .XmitAsync
         Message  " C:STOP "
```

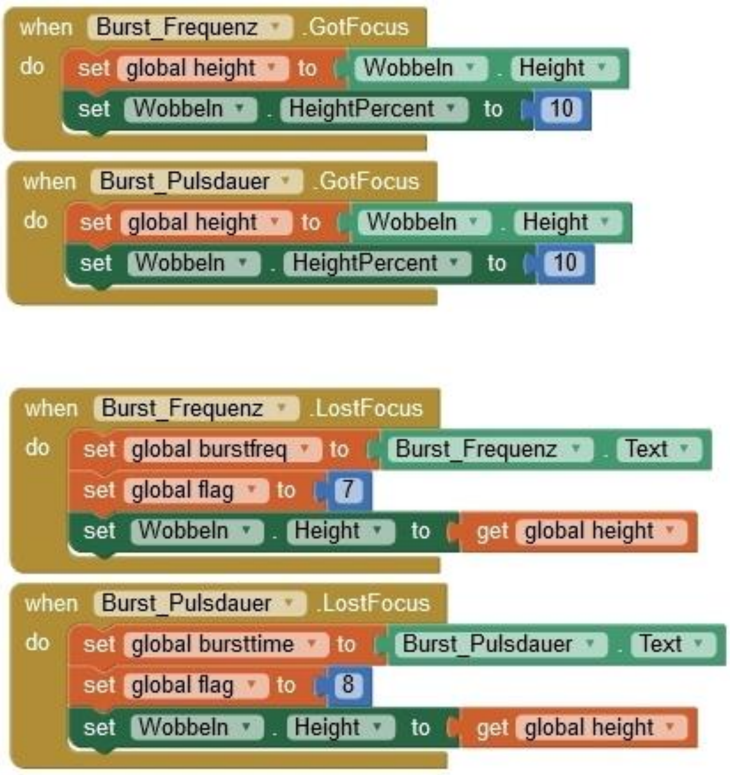
Abbildung 16: Dauerton-Steuerblock 1



```
when XMIT_Continuous .GotFocus
do  call XMIT_Continuous .HideKeyboard
    if      get global flag = 5
    then    call UDPXmitter1 .XmitAsync
                          Message  join  " C:F: "
                                         get global freq
            set Eingabe . Text to  get global freq
    else if get global flag = 6
    then    call UDPXmitter1 .XmitAsync
                          Message  join  " C:D: "
                                         get global duty
            set Eingabe . Text to  get global duty
    else    set global flag to 0
    set global flag to 0
```

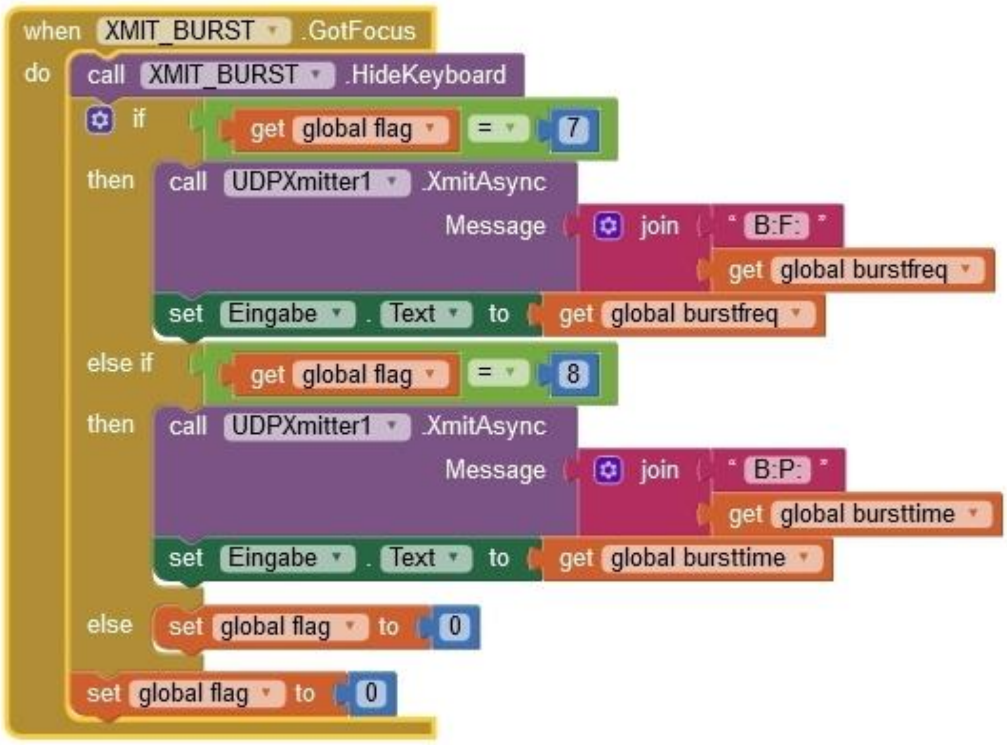
Abbildung 17: Dauerton-Steuerblock 2

Entering values in the burst range is only feasible with a trick. When you tap an input field, it disappears behind the keyboard. If a number is entered blindly, the field is still empty afterwards, the number has passed into nirvana. Now you could take a second screen or, as I have solved that, push the upper part together so that the burst area has enough space. This is exactly what happens, as soon as one of the text fields in this area receives the focus, the horizontal arrangement wobbling goes to the height of 10%. The input is processed in the same way as in the other two areas.



```
when Burst_Frequenz .GotFocus
do  set global height to  Wobbeln . Height
    set Wobbeln . HeightPercent to 10

when Burst_Pulsdauer .GotFocus
do  set global height to  Wobbeln . Height
    set Wobbeln . HeightPercent to 10

when Burst_Frequenz .LostFocus
do  set global burstfreq to  Burst_Frequenz . Text
    set global flag to 7
    set Wobbeln . Height to  get global height

when Burst_Pulsdauer .LostFocus
do  set global bursttime to  Burst_Pulsdauer . Text
    set global flag to 8
    set Wobbeln . Height to  get global height
```

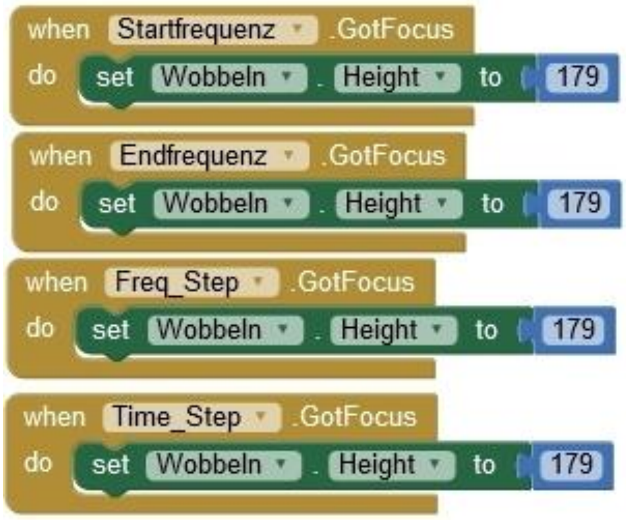
Abbildung 18: Burst-Steuerung 1



```
when XMIT_BURST .GotFocus
do  call XMIT_BURST .HideKeyboard
    if      get global flag = 7
    then    call UDPXmitter1 .XmitAsync
                         Message  join  " B:F: "
                                        get global burstfreq
            set Eingabe . Text to  get global burstfreq
    else if get global flag = 8
    then    call UDPXmitter1 .XmitAsync
                         Message  join  " B:P: "
                                        get global bursttime
            set Eingabe . Text to  get global bursttime
    else    set global flag to 0
    set global flag to 0
```

Abbildung 19: Burst-Steuerung 2

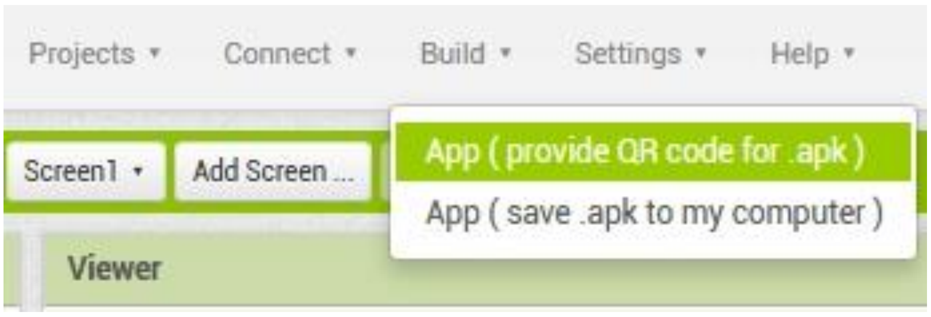
Of course, the wobble area must be enlarged again if entries are to be made there. Because there is no way to set the value for the height back to automatic, as in the design, you have to specify the height as a numerical value.



```
when Startfrequenz .GotFocus
do  set Wobbeln . Height to 179

when Endfrequenz .GotFocus
do  set Wobbeln . Height to 179

when Freq_Step .GotFocus
do  set Wobbeln . Height to 179

when Time_Step .GotFocus
do  set Wobbeln . Height to 179
```

Abbildung 20: Wobbeln groß schalten

If we don't have any warnings or error messages, let's start compiling the app.


Projects
Connect
Build
Settings
Help
App (provide QR code for .apk)
App (save .apk to my computer)
Screen1
Add Screen ...
Viewer
Abbildung 21: Build App and install

We can either request a QR code so that the app can be downloaded directly to the mobile phone after completion, or we can save the apk file on the PC so that we can later transfer or share it, for example via Bluetooth. To download the app, please follow the instructions on page 20. There is no listing in text form for this part of the project, it is replaced by the illustration of the blocks.

I wish you happy repelling mosquitoes and marten.