Heute zeige ich Ihnen, wie man eine App für ein Androidgerät, Handy oder Tablett, ganz einfach mit Hilfe grafischer Elemente entwickeln kann, um damit zum Beispiel das Robot Car zu steuern. Herzlich willkommen zur neuen Folge von

Micropython auf dem Robot Car
Teil 3 - Wir steuern das Auto mit dem Handy

Das Handy erkennt die Lage im Raum und stellt zum Beispiel die Bildschirmdarstellung je nach Ausrichtung des Geräts automatisch um. Dahinter steckt ein Verwandter des MPU6050-Chips, der auf dem Modul GY521 verbaut ist, das wir in der ersten Folge in unsere Handsteuerung eingebaut haben. Vielleicht haben Sie dieses Feature bereits in der zweiten Folge zusammen mit Ihrem Robot Car ausprobiert, was rede ich, natürlich haben Sie das schon getan. Nun, was mit der Handsteuerung funktioniert, können Sie nach Durchführung der Schritte in dieser Beschreibung auch mit Ihrem Handy machen. Keine Sorge, es wird nicht kompliziert. Wenn Sie in jungen Jahren gerne mit Bauklötzen gespielt haben, meistern Sie das problemlos.

Besorgen und Installation der Software

Die Hardware ist im Prinzip komplett, es sei denn sie wollen Ihrem Auto noch zwei LEDs für die Rücklichter spendieren. Was Sie jetzt und hier brauchen, sind demnach nur zwei Dinge, das Robot Car und Ihr Android-Smartphone. Na ja, und dann ist da noch die Software. Um die freie Software vom MIT (Massachusetts Institute of Technology), das auch Lizenzinhaber ist, zu nutzen, gibt es drei Wege. Zwei davon führen über Softwareteile, die auf dem PC installiert werden müssen. Ich habe die dritte Möglichkeit gewählt, die auch von den Machern von App-Inventor 2 (AI2) vordringlich empfohlen wird. Auf dem PC kommt dazu nur ein Browser zum Einsatz und auf dem Handy die App **MIT AI2 Companion**. Auf dem PC tut's ein Browser wie **Firefox** oder **Chrome**. Mit Internet Explorer funktioniert es definitiv nicht, aber der ist eh tot. Ob Edge einsetzbar ist, habe ich nicht getestet, wozu auch? Den MIT AI2 Companion können Sie über den Google Playstore installieren. **Damit diese Vorgehensweise korrekt arbeitet muss das Handy in einem lokalen Funknetz eingeloggt sein, nicht über das Internet**. Ein Zugriff über das Internet kann nicht funktionieren, weil vom Handy aus zwar beliebige Ziele erreichbar sind, umgekehrt das Handy aber vom Internet aus nicht direkt angesprochen werden kann. Es ist im lokalen Netz des Providers gekapselt und deshalb von außen nicht sichtbar. Das heißt, der nötige Zugriff des Browsers auf das Handy aus dem Internet ist nicht möglich, weil keine öffentliche IP und kein Routing existieren.

Das grundsätzliche Werkzeug steht nun bereit. Es gibt aber ein kleines Handicap, das Handy kann zwar perfekt den TCP-Dialekt, versteht aber erst einmal kein Wort UDP. Das ist für die Standardanwendungen auch nicht erforderlich. Also verpassen wir ihm via AI2 einen kleinen Sprachkurs. Den habe ich auf Ullis Roboterseite gefunden. Hier ist die Downloadadresse:

https://ullisroboterseite.de/android-AI2-UDP/UrsAI2UDP.zip

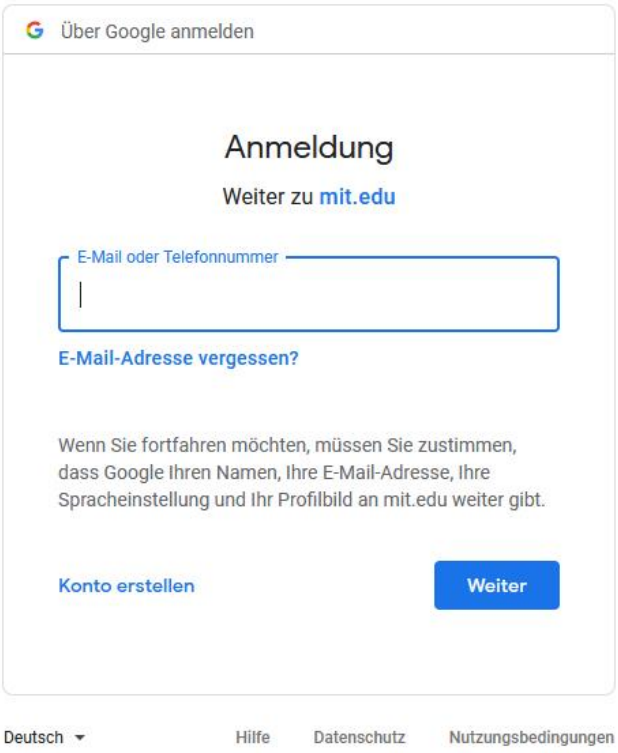
Entpacken Sie das Archiv zunächst in ein beliebiges Verzeichnis. Wechseln Sie dort hin und überzeugen Sie sich, dass die Datei **de.ullisroboterseite.ursai2udp.aix** aufgelistet ist. Sie enthält die Erweiterungen, die wir zur Erzeugung unserer Steuer-App in Kürze brauchen.

App Inventor 2 erstmals starten

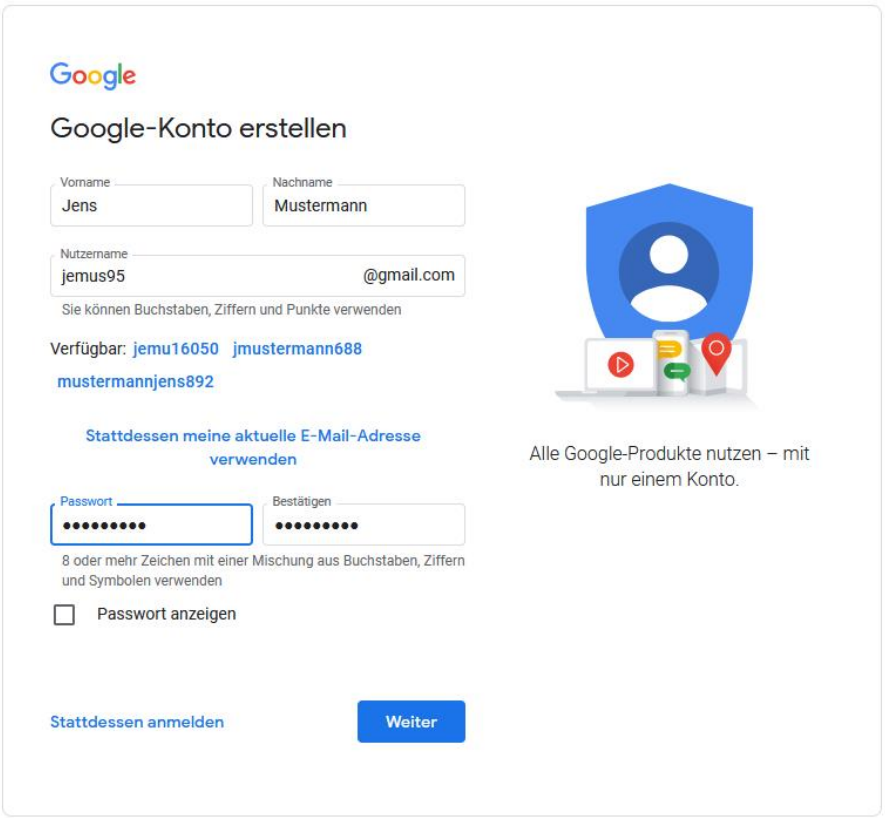
Öffnen Sie jetzt Chrome oder Firefox, falls noch kein Browser läuft und geben Sie folgende URL ein.

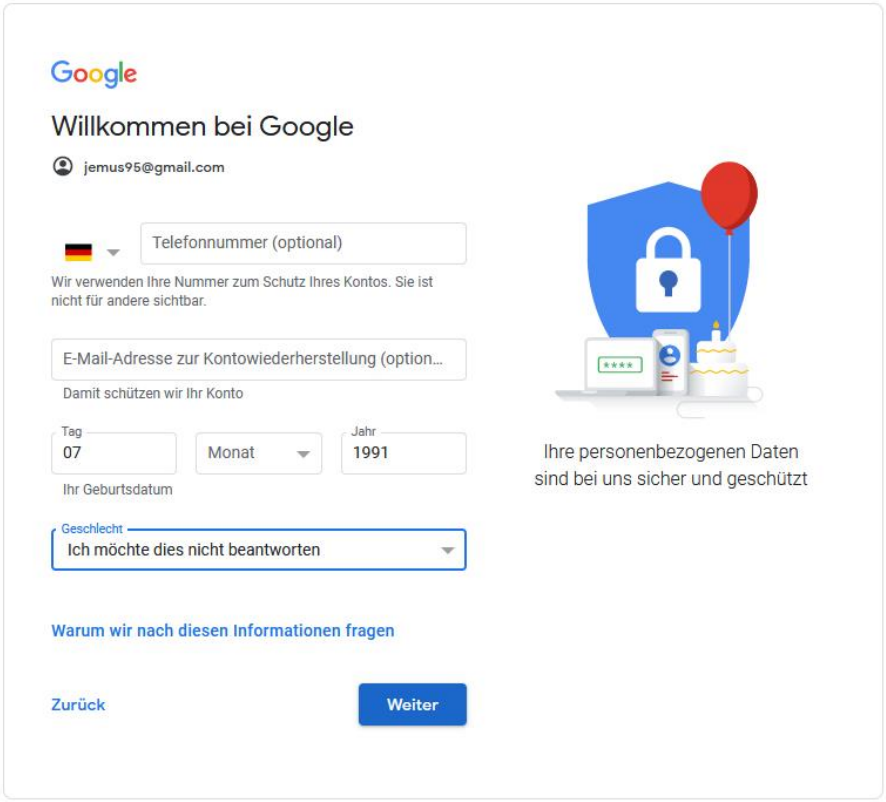
http://ai2.appinventor.mit.edu

Die Anmeldung bei Appinventor ist nur über ein Googlekonto möglich. Haben Sie bereits eines, dann klicken Sie auf **Weiter**, sonst auf **Konto erstellen**.

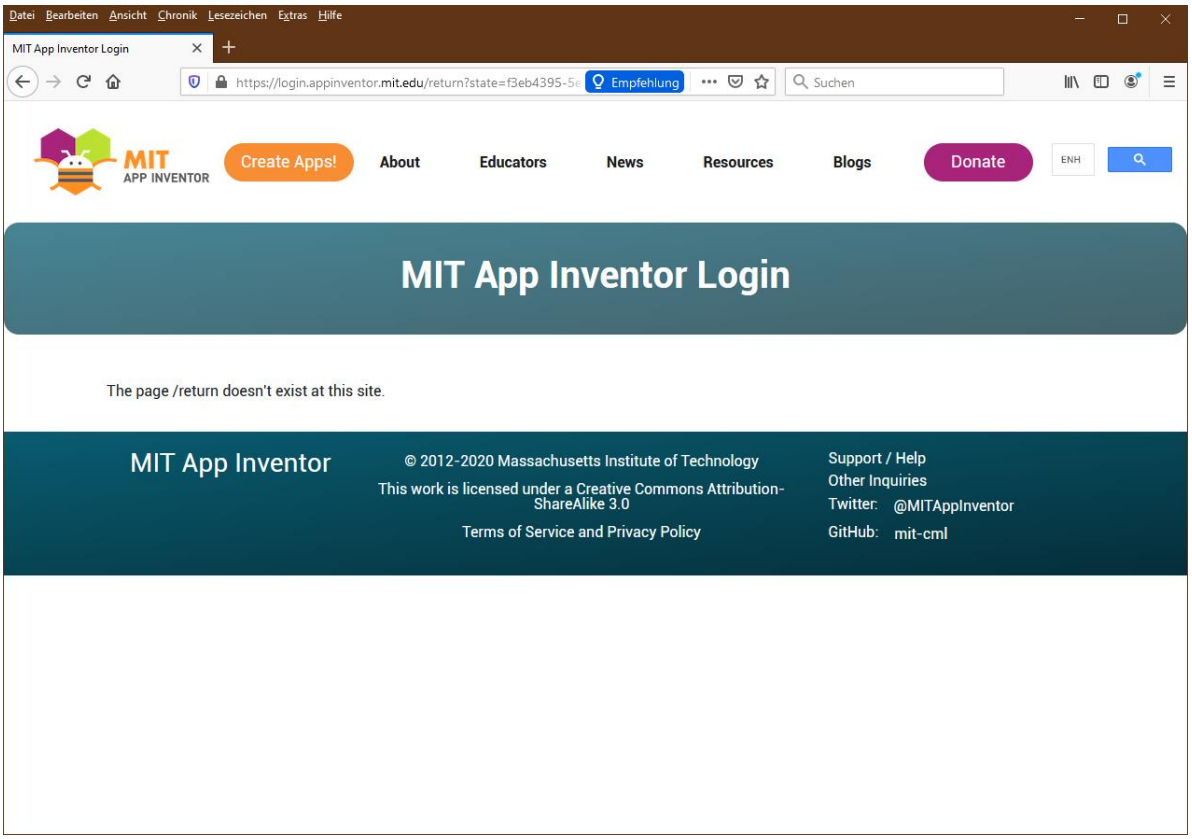


Das Konto kann auch mit gefakten Daten angelegt werden. Das gilt für die erste und auch für die folgende Seite.



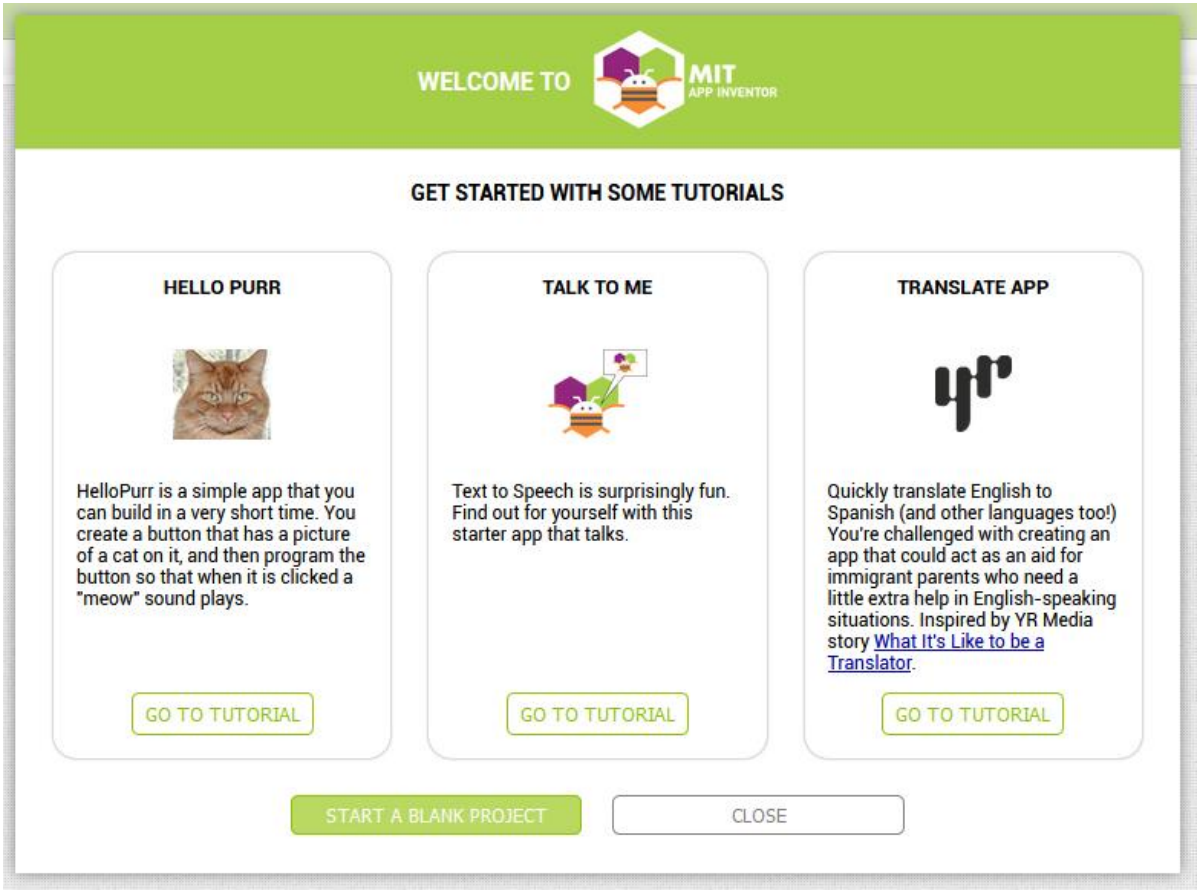


Nach den üblichen Einstellungen zur Privatsphäre landen Sie auf dieser Seite.

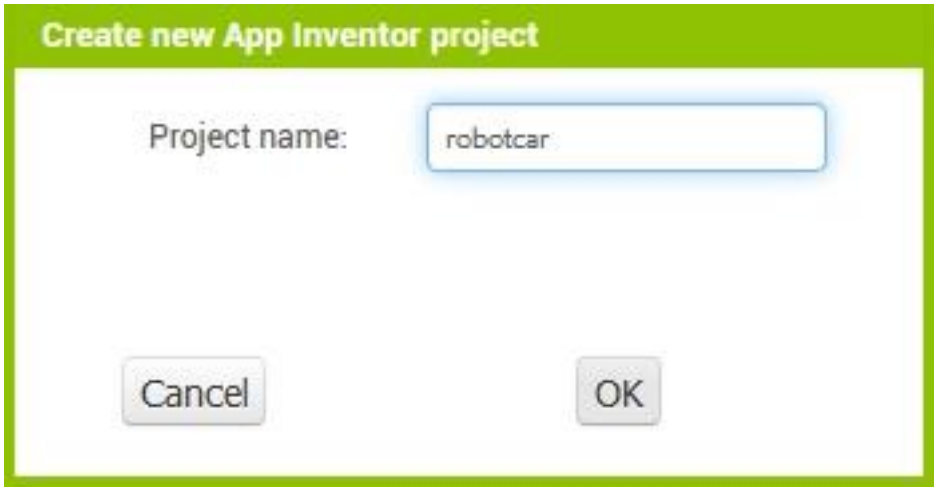


Der MIT APP INVENTOR -Link führt Sie zurück zum Login. Melden Sie sich mit Ihrem Googlekonto an und lassen Sie sich nicht von dem Riesenangebot an Tutorials und

Beispielen verführen, sondern starten Sie einfach ein neues, leeres Projekt, Sie wollen doch das Robot Car mit einer eigenen App steuern, oder? Natürlich wollen Sie das!

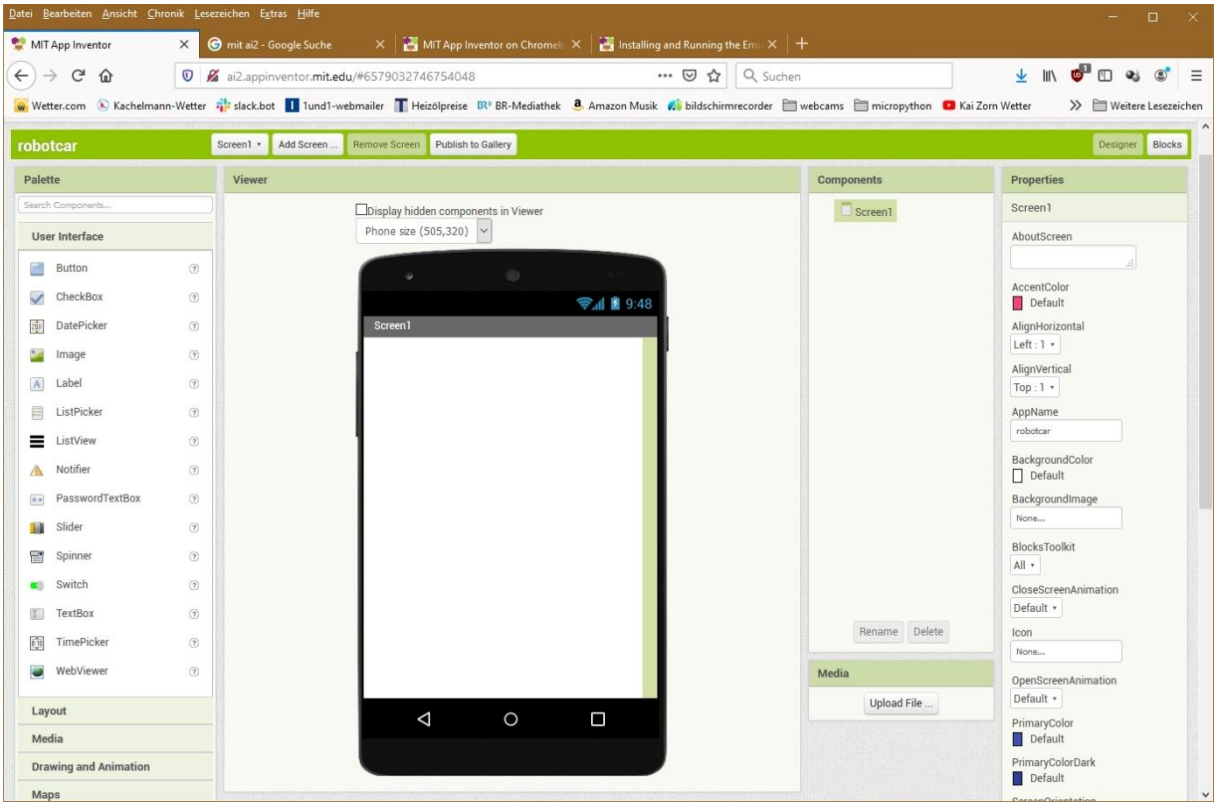


Ich weiß aber auch, dass Sie später begierig alles aufsaugen werden, was AI2 Ihnen bietet - mir ging es jedenfalls so.



Einrichtung der Bildschirmobjekte

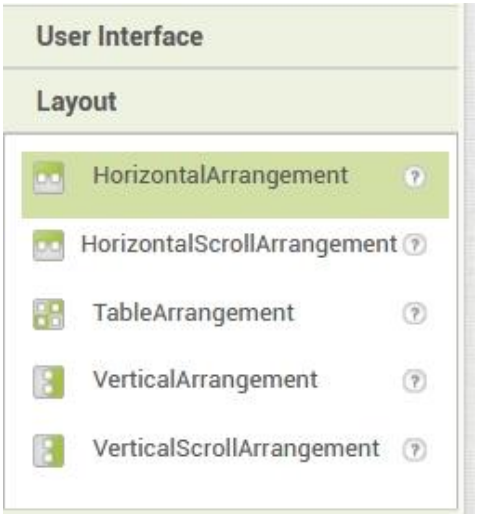
Werfen wir einen Blick auf die Seite auf der das Layout der App entworfen wird. In dem Abschnitt **Palette** finden Sie die Zutaten. Der mittlere Bereich, **Viewer**, ist Ihre Arbeitsfläche. Rechts davon in der Sektion **Components** wird die Hierarchie der verwendeten Komponenten dargestellt und rechts daneben, in **Properties**, die Eigenschaften der gerade ausgewählten Komponente. Das alles sehen Sie, wenn sie in der **grünen Zeile** über diesen Bereichen rechts außen **Designer** ausgewählt haben.



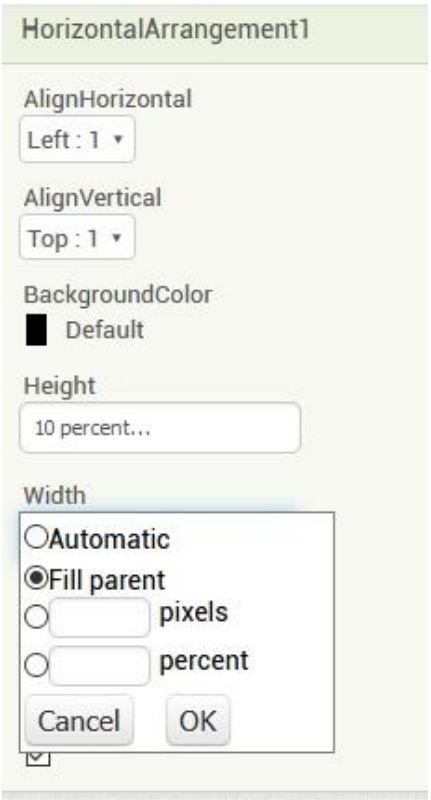
Wir werden jetzt Stück für Stück zum Layout hinzufügen und die Eigenschaften der Komponenten einstellen. Alle Eigenschaften, die nicht explizit aufgeführt sind, bleiben auf Defaulteinstellung.

Die Screenkomponente ist bereits aktiviert. Wir geben ihr den Titel ROBOTCAR STEUERUNG. Scrollen Sie in Properties nach unten und geben Sie bei Title den Text ein.

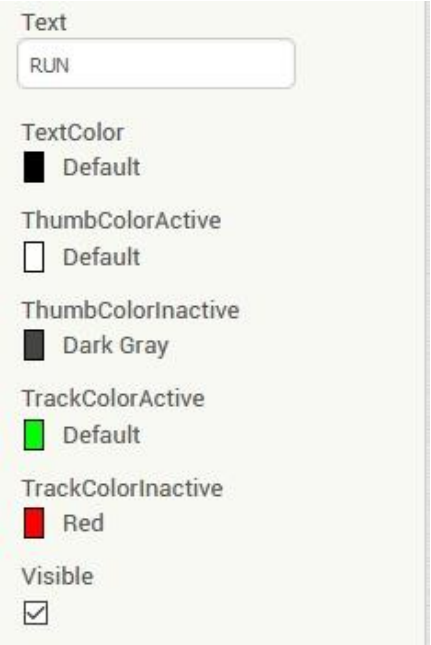
In der ersten Zeile sollen ein Schalter und ein Textfenster erscheinen. Also holen Sie aus dem Ordner **User Interface** einen **Switch** und ein **Label** und ziehen sie die beiden auf die Handyfläche. Was, das Label will nicht neben dem Switch wohnen. Geben Sie den beiden halt ein ordentliches Zuhause, holen Sie sich aus dem Ordner **Layout** ein **HorizontalArrangement** und setzen Sie es über den Schalter. Jetzt können Sie die Elemente Switch1 und Label1 hineinsetzen und schauen Sie, jetzt nehmen sie auch friedlich neben einander Platz.



Verteilen wir die Eigenschaften. Markieren Sie in Components den Eintrag **HorizontalArrangement1**. Klicken Sie auf das Feld von **Width**, **Fill Parent** und **OK**.



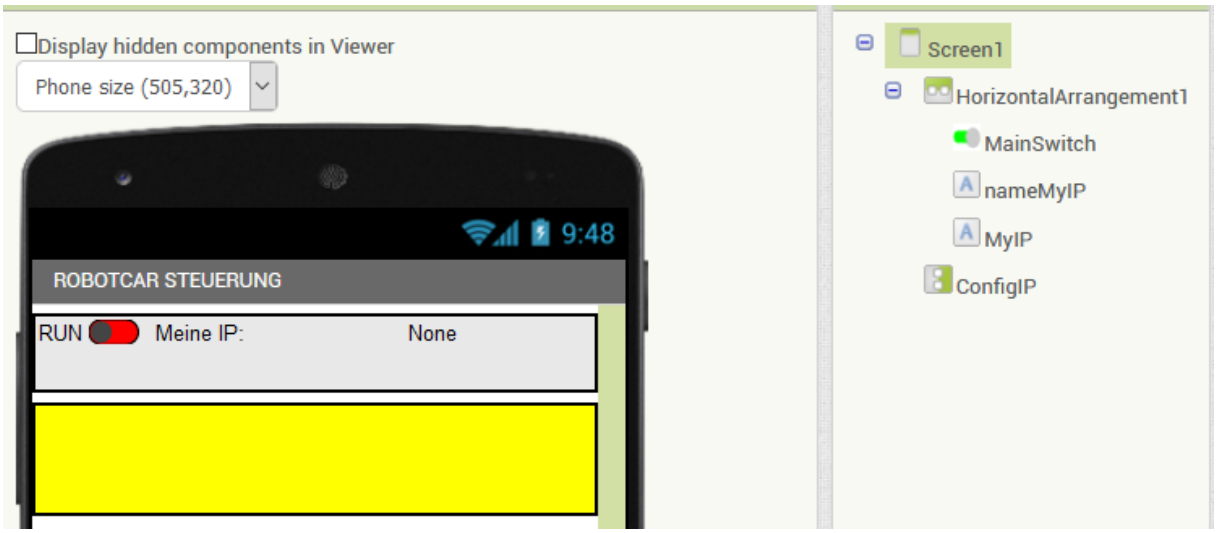
Markieren Sie in **Components Switch1**. Linksklick auf **Rename,** dann geben Sie den neuen Namen **MainSwitch** ein. Ändern Sie die Eigenschaften wie folgt.



Benennen Sie **Label1** in **MyIP** um und stellen Sie die Eigenschaften auf folgende Werte ein.

Fügen Sie zwischen **MainSwitch** und **MyIP** ein weiteres Label mit dem Namen **nameMyIP** und dem Text **Meine IP** ein.

Das nächste Arrangement heißt **ConfigIP**, ist vertical, der Hintergrund gelb, die Höhe 20% des Displays und es nimmt die gesamte Bildschirmbreite ein. Ich denke, Sie kriegen das hin!?



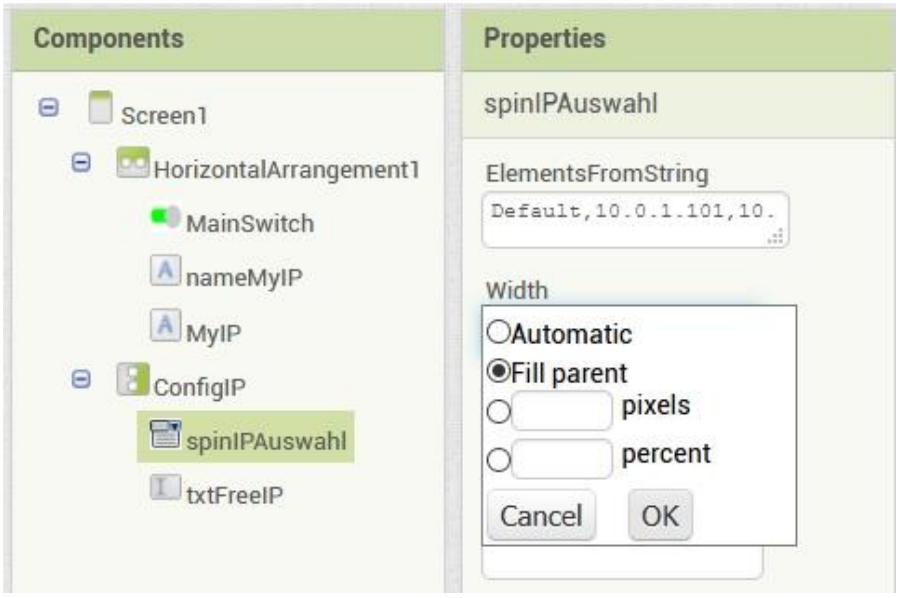
Jetzt zur Einrichtung. Ein **Spinner** ist eine Dropdownliste, aus der Einträge durch Touch ausgewählt werden können. Die Einträge werden durch eine Liste festgelegt. Die Begriffe sind durch Kommas getrennt. Beim Bildschirmaufbau ist der erste Begriff bereits markiert. Eine **Textbox** ist ein Eingabefeld, dessen Inhalt beim Verlassen übernommen wird.

Wir legen eine Liste von IP-Adressen fest, die mögliche Ziele des Projekts darstellen, WLAN-Accesspoint, ESP32-Accesspoint und PC zu Testzwecken. Die Liste sieht aus wie folgt: Klar, dass Sie die IP-Adressen an Ihre Netzumgebung anpassen müssen.

Default,10.0.1.101,10.0.2.101,10.0.1.10

Die ausgewählte Adresse soll zur Laufzeit in der Textbox dargestellt werden, um die Auswahl mitzuteilen und /oder um Änderungen zu ermöglichen.

Die Eigenschaften des Spinners sind hier dargestellt.



Finden Sie die Eigenschaften der Textbox **txtFreeIP** heraus? Die Schriftgröße verrate ich Ihnen: 18.

Haben Sie alles? BackgroundColor: Orange, FontSize: 18, Width: Fill Parent, Text: default, TextAlignment: center 1, TextColor: Blue.

Es folgt jetzt ein **TableArrangement** mit 2 Spalten und 2 Zeilen. Ich nenne es DriveData. Es füllt die ganze Bildschirmbreite und enthält 4 Lables, die zwei linken, **txtVelo** und **txtDir**, zur Beschreibung und die rechten, **lblVelo** und **lblDir**, für die Wertangabe von Geschwindigkeit und Fahrtrichtung.

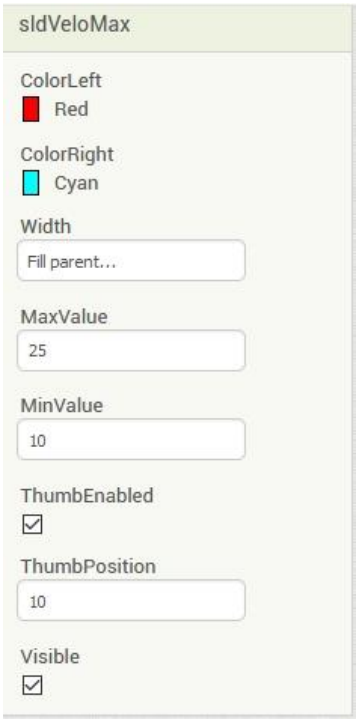
txtVelo und txtDir: FontSize: 14, Textalignment: right 2, Width: 50% Text: Geschwindigkeit | Richtung

lblVelo und lblDir: FontSize: 20, FontBold, Text: 0

Im nächsten Arrangement (horizontal, height: 20%, Width: Fill Parent) finden zwei Buttons Platz, btnLight und btnMotor. Für die Buttons gilt: BackgroundColor: Green, FontSize: 30, Shape: rounded, TextAlignment: center 1, Text LICHT | MOTOR.

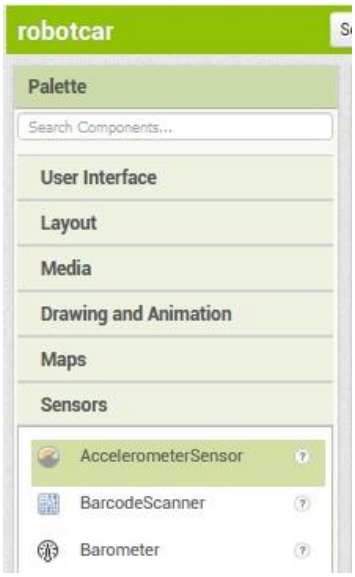
In einem anschließenden HorizontalArrangement, VeloMaxDisplay (Width: Fill Parent), wohnen ein Beschreibungslabel, txtVeloMax (Width: 50%, TextAlign: center 1, Text: Maximale Geschwindigkeit) und ein Label lblVeloMax, das den gerundeten Wert des folgenden Schiebereglers darstellt.

Den Abschluss bildet ein Slider (aka Schieberegler) mit Namen sldVeloMax mit folgenden Properties:



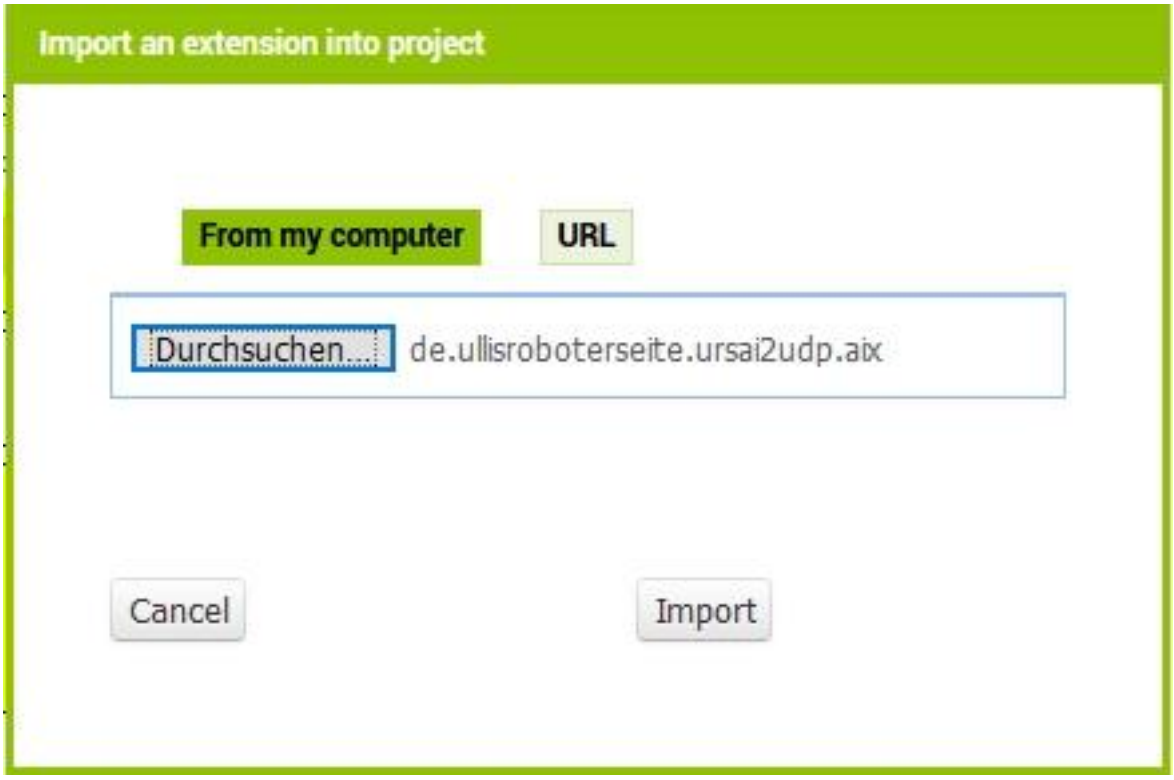
Unser Display sieht nun so (ähnlich) aus.

Jetzt fehlen uns noch zwei "unsichtbare" Elemente, ein Accelerometer und der UDP-Sender. Den Accelerometer Sensor finden Sie im Palette-Ordner **Sensors**. Ziehen Sie ihn auf die Handyfläche. Er wird unterhalb derselben angezeigt.



Wissen Sie noch in welchem Ordner sich die Datei **de.ullisroboterseite.ursai2udp.aix** befindet? Diese Datei brauchen wir als Nächstes. Wir müssen damit den Sprachumfang von AI2 erweitern. Rollen Sie in **Palette** ganz nach unten, klicken Sie dort auf **Extension** und dann auf **Import extension**.

Im folgenden Fenster navigieren Sie nach Klick auf **Durchsuchen** zur aix-Datei.



Nach Klick auf **Import** werden nach ein paar Sekunden zwei Einträge angezeigt. Ziehen Sie UDPXmitter auf die Handyfläche. Jetzt beherrscht Ihr Handy auch den UDP-Dialekt und kann Datagramme an den Server auf dem Robot Car senden. Wenn Sie dieses Feature auch in anderen Projekten nutzen möchten, müssen Sie den Importvorgang jedes Mal wiederholen, nachdem das neueProjekt angelegt wurde.

Die Programmierung

Wenden wir uns nun der Programmierung zu. Bis jetzt haben wir ja nur Daten und Objekte bereitgestellt. Jetzt erwecken wir das Ganze zum Leben, jetzt dürfen Sie Bausteine stapeln. Wechseln Sie in der grünen Menüleiste mit dem Button **Blocks** rechts außen ins Programmierfenster von AI2.

Links ist der Bereich für Programmblöcke oder Bausteine mit eingebauten Blocks für Text, Logik, Datenstrukturen, mathematischen Funktionen und Operatoren, Variablen und Proceduren. Darunter finden Sie die Blocks der Elemente, die wir bisher festgelegt haben.

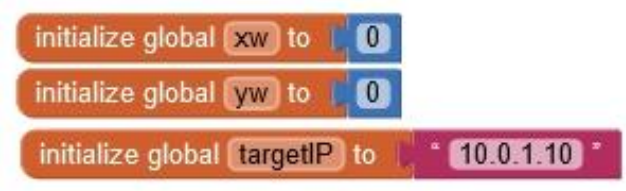
Im Viewerfenster bauen wir jetzt für verschiedene Events aus den Bausteinen unsere "Häuser". Sie können nur Bausteine zusammenstecken, die auch zusammenpassen. Das entspricht den Syntax- und Semantikregeln von MicroPython und weiteren Programmiersprachen. Blocks wie in AI2 sind Sie vielleicht schon mal begegnet. Tools wie Squeak und Scratch basieren auf der gleichen Methode zu programmieren. Dahinter steckt die Sprache Smalltalk-80.

Zugegeben, die Sache ist etwas holprig für Leute, die gewohnt sind, Programmtexte frei weg zu schreiben. Aber mal ehrlich, wie oft vertippen Sie sich und suchen dann nach den Fehlern? Wenn Fehler in Scratch oder AI2 auftreten, dann sind das logische Fehler, aber keine Syntaxfehler. Daher ist diese Art zu programmieren für Anfänger gut geeignet und wird in Schulen gern praktiziert.

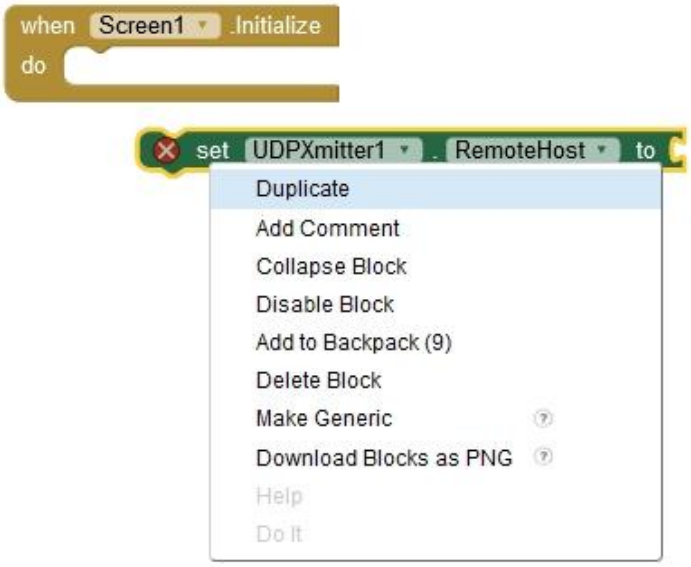
Wir bereiten schon mal verschiedene Variablen vor, dann initialisieren wir den Viewport. Ich werde zu Beginn alles sehr genau beschreiben. Im weiteren Verlauf werden dann die Angaben spärlicher und beschränken sich schließlich nur noch auf wesentliche Dinge wie die Auflistung der Properties. Die Farben der verschiedenen Bereiche helfen bei der Orientierung. Unbenannte Strukturblöcke stammen aus dem Built-In-Bereich, die Beschriftung der benannten Blöcke verrät ihre Herkunft.

Ziehen Sie aus **Variables** ein **initialize global name to** in den **Viewer**. Hängen Sie aus **Math** eine 0 dran. Klicken Sie auf **name** und geben Sie **xw** ein, Enter. Widerholen Sie den Vorgang für **yw**.

Erzeugen Sie eine globale Variable **targetIP** und hängen Sie aus Text ein **" "** dran. Geben Sie dann die IP-Adresse Ihres PCs ein. Zur Demonstration des Einfügens und Verklebens von Bausteinen sollte das genügen. Wir ergänzen später weitere Variablen

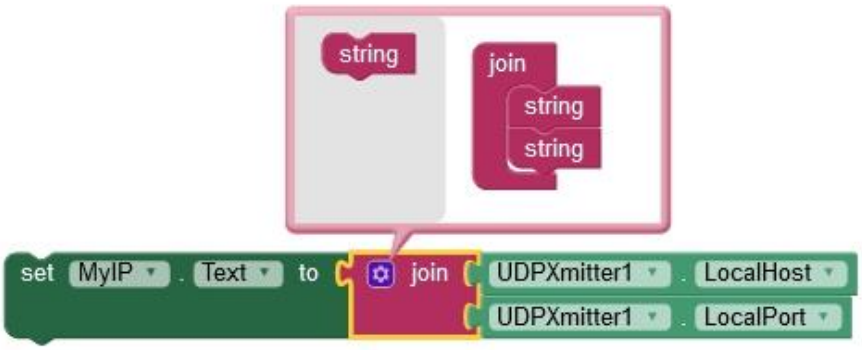


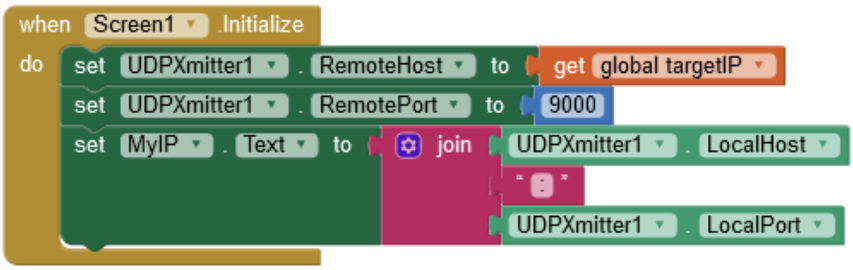
Ziehen Sie jetzt aus **Screen1** eine **when Screen1 initialize**-Klammer.



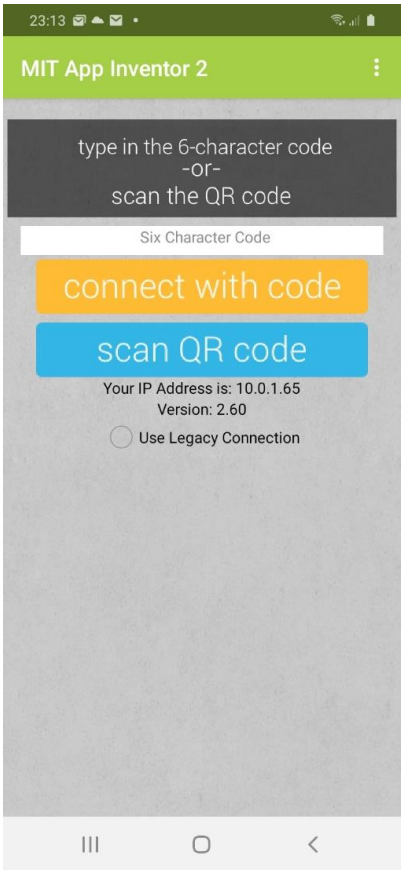
Ein Rechtsklick auf den set UDPXmitter1-Streifen aus **UDPXMitter1** und ein Klick auf **Duplicate** erzeugen ein Duplikat des Blocks. Verkleben Sie den zweiten mit dem ersten und Ändern Sie im zweiten Block RemoteHost in **RemotePort**. Aus Variables holen Sie ein **get** und aus Math eine 0. Kleben Sie das get an RemoteHost und die 0 an RemotePort. Ändern Sie die 0 in 9000 und im get-Block wählen Sie targetIP aus.

Aus **MyIP** holen Sie ein **set MyIP.Text to** und kleben Sie ein **join** aus Text dran. Kleben Sie aus UDPXmitter1 ein **LocalHost** und **LocalPort** ans **join**. Für den trennenden Doppelpunkt brauchen Sie eine weitere Kontaktstelle. Ein Klick auf das blaue Symbol öffnet das Fenster für die Einstellung. Ziehen Sie **string** aus der linken Hälfe zwischen die beiden in der rechten Hälfte und ergänzen Sie dann den Doppelpunkt.

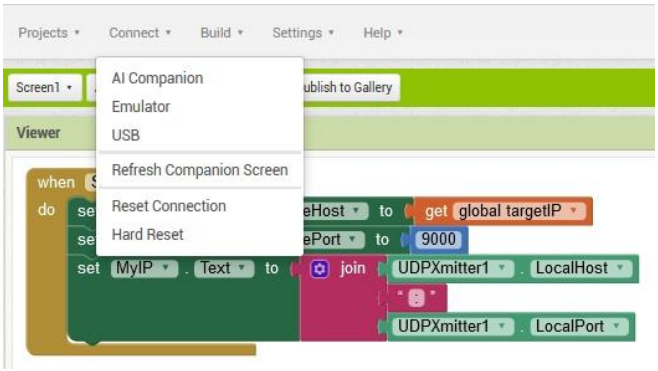




Wenn Sie keine Fehlermeldungen und Warnungen haben, sind Sie sicher gespannt darauf, wie das alles auf Ihrem Handy aussieht. Verbinden Sie das Handy mit dem WLAN-Router und starten Sie auf dem Handy den AI2-Companion.



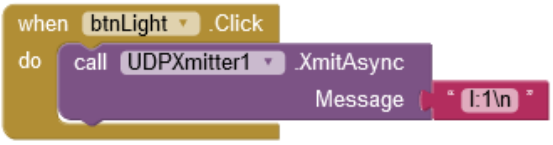
Besonders einfach geht die Kontaktaufnahme zwischen Handy und PC über den QR-Code-Scan. Auf PC-Seite starten Sie den Vorgang mit Klick auf **Connect** und **AI-Companion**.

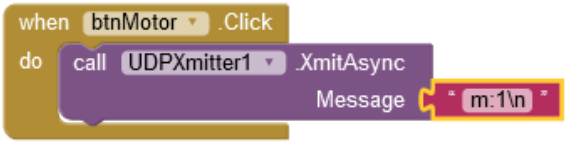


Scannen Sie jetzt den QR-Code mit dem Handy ein.

Der Progressbar läuft durch und danach haben Sie Ihren Entwurf mit dem ersten Programmierergebnis auf dem Handydisplay. Die lokale IP des Smartphones wird rechts neben Meine IP angezeigt. Sie können auch schon den Schalter antippen, die Spinnerliste aufklappen und den Schieberegler bedienen. Damit dadurch auch Aktionen ausgelöst werden, müssen wir noch eine Menge Code ergänzen. Übrigens, nicht erschrecken, die Verbindung zum Handy wird relativ schnell wieder gekappt, wenn keine Aktionen erfolgen. Dadurch wird nichts gelöscht oder verändert, Sie müssen nur wieder neu verbinden. Wenn Sie später die komplette App auf dem Handy installieren, gibt es auch kein automatisches Trennen der Verbindung mehr.

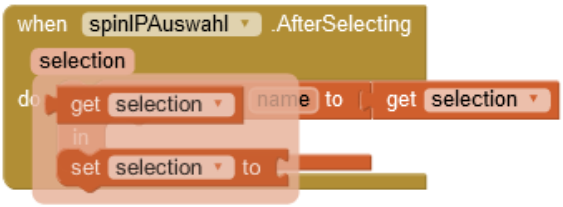
So bringen Sie Leben in die Buttons für das Fahr-Licht und das Motor-Relais.



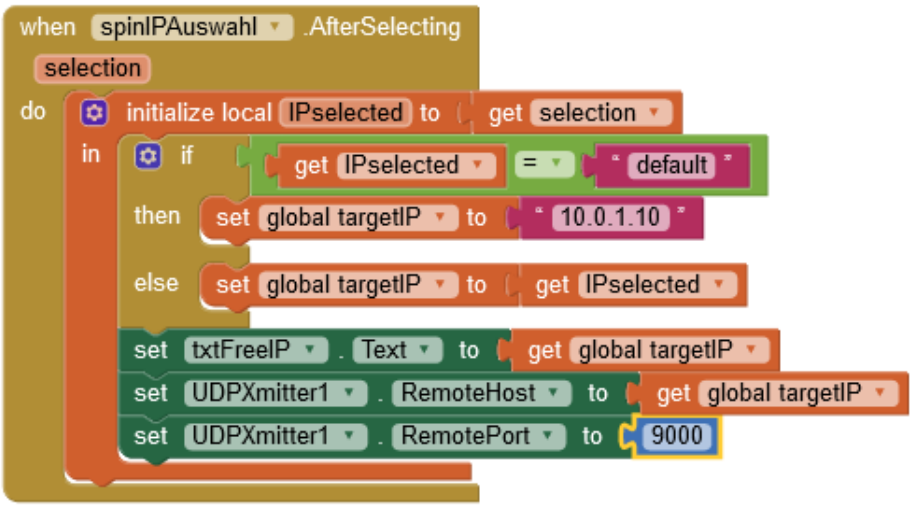


Wie die Namen der when-Klammern erkennen lassen, stammen sie aus dem Blockvorrat der Buttons btnLight und btnMotor. die Functioncalls stammen aus UDPXmitter1. Die Kommandostrings müssen mit einem Linefeed abgeschlossen werden, das erledigt das "\n".

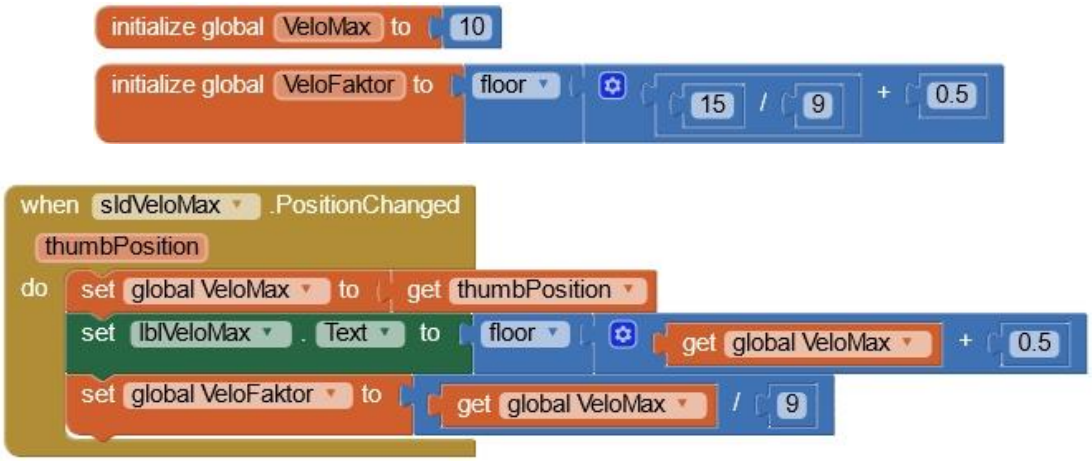
Jetzt geht es dem Spinner an den Kragen. Für die Behandlung der Auswahl erzeugen wir eine lokale Variable **IPselect** und weisen aus dem **selection**-Feld **get selection** zu.



Der Rest der "Füllung" erklärt sich von selbst.



Für die Ermittlung der Fahrstufen brauchen wir eine Funktion, welche die Floating-Rohwerte vom Accelerometer in brauchbare Integerwerte im zulässigen Bereich umwandelt. Zum Einstellen der Bereichsobergrenze dient der Slider. Damit man den eingestellten Wert auch vor Augen hat, weise ich **lblVeloMax.Text** den gerundeten Wert des Schiebereglers zu. Für die Berechnungen brauche ich zwei neue globale Variablen.



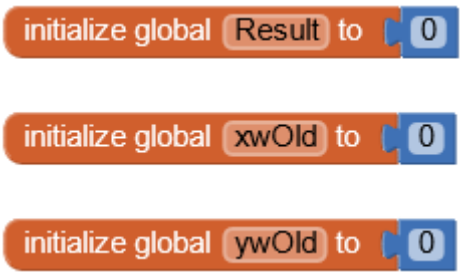
Für die Berechnung der Fahrstufen benötige ich das Verhältnis aus der maximal möglichen Fahrstufe und den maximalen Absolutwerten des Accelerometerchips. Diese habe ich zu 9 in alle Richtungen gemessen. Der berechnete Wert steht in **VeloFaktor** bereit.

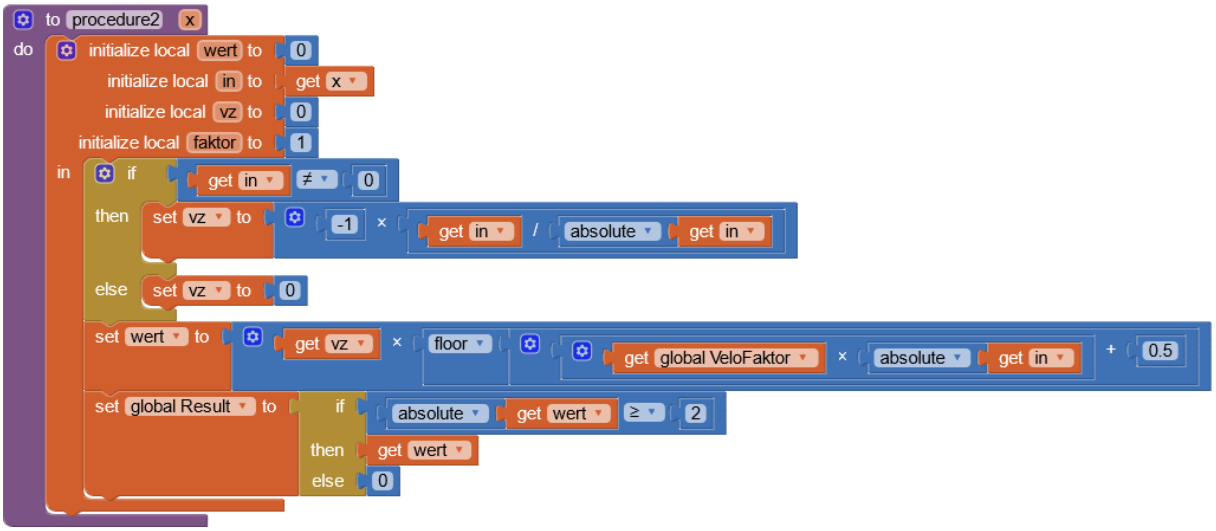
So wie die Rohwerte vom Accelerometer kommen, sind die Vorzeichen für beide Achsen komplementär. In der Funktion **procedure2** wird daher nach der Einführung von einigen lokalen Variablen das Signum **vz** des Parameters x berechnet und, falls ungleich Null, invertiert.

Die Fahrstufe **wert** wird nach folgender Formel berechnet:

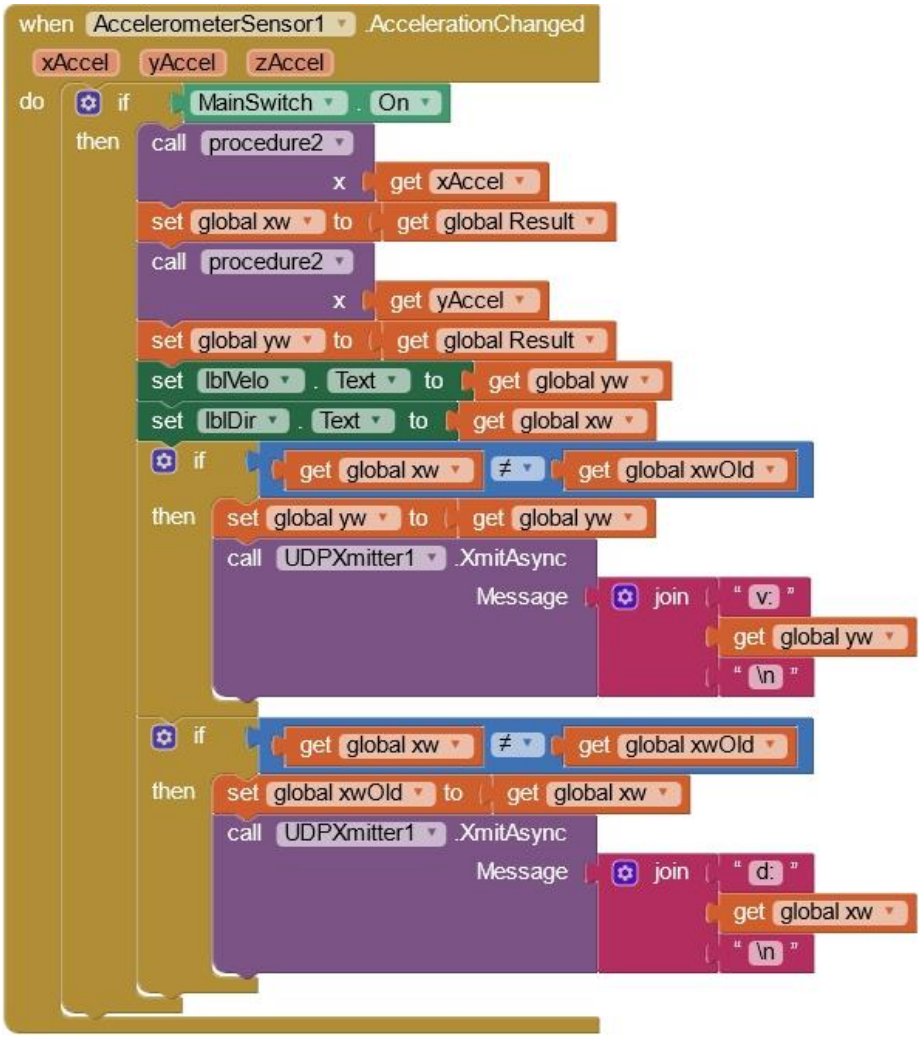
wert =vz*int((VeloFaktor * abs(x))+0,5)

Für die Rückgabe des Ergebnisses dient die neue globale Variable Result. Damit das in einem Rutsch geht definieren wir auch gleich noch zwei Merker für xw und yw aus dem vorherigen Durchgang.



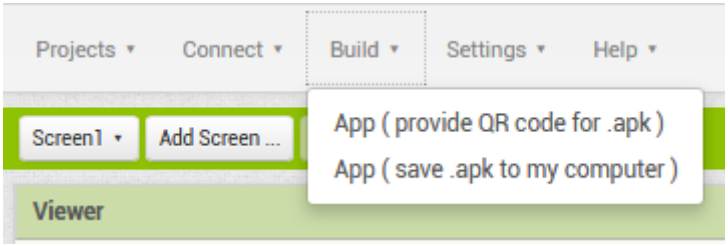


Damit sind wir auch schon beim letzten Block angelangt. Hier werden die Acceleratorwerte erfasst, auf Fahrstufen umgerechnet und falls sich ein Wert geändert hat, wird er an das Robot Car gesendet. Damit das alles auch wirklich passiert, muss der Schalter **MainSwitch** natürlich an sein.



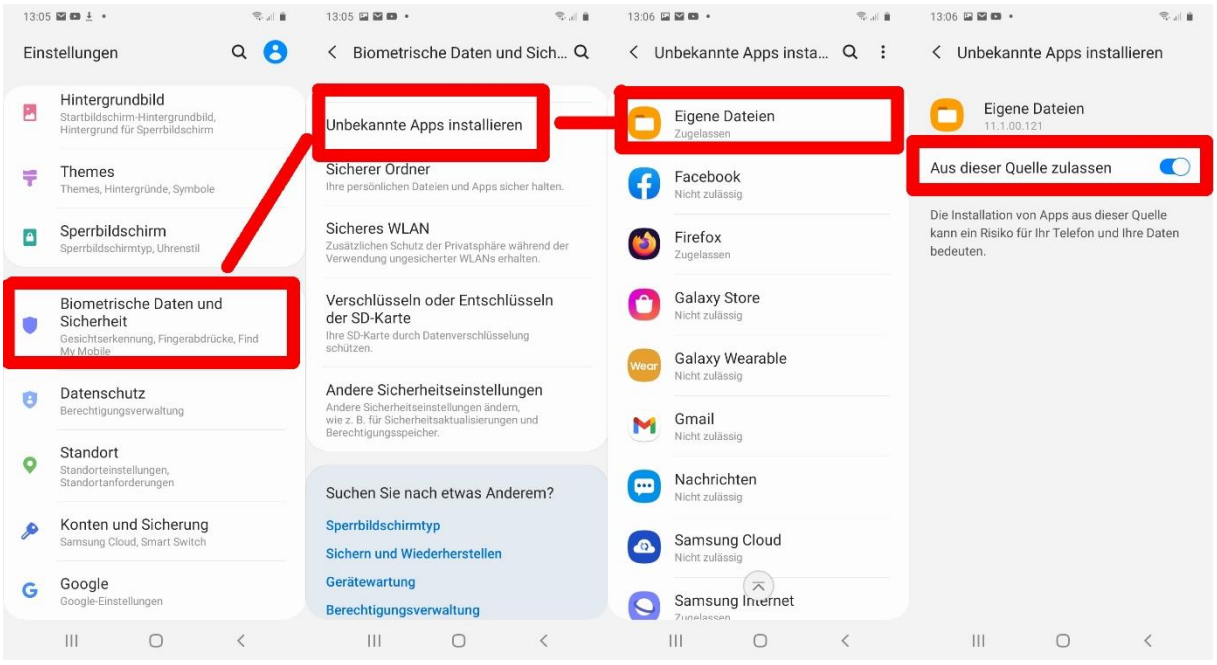
Jetzt ist alles im Kasten und damit ist es an der Zeit, den finalen Funktionstest am Handy zu absolvieren. Starten Sie das Robot Car und verbinden Sie Handy und PC, wie Sie es oben schon mal gemacht haben. Mainswitch an, Handy nach vorne neigen und los geht's. Fahren Sie ein paar Kurven, wenden Sie – perfekt!

Dann lassen Sie uns aus dem Projektentwurf eine dauerhafte App machen, die Sie auf dem Handy installieren können. Compiliert wird die Anwendung in jedem Fall in der AI-Cloud. Dort wird Ihr Projekt auch gespeichert und von dort wird die apk-Datei heruntergeladen. Der Button **Build** bietet zwei Möglichkeiten an. Mit der ersten Option wird nach ca. 1 bis 2 Minuten nach dem Anklicken ein QR-Code angezeigt. Er enthält die Download-URL. Scannen Sie ihn ein. Nach dem Download startet die Installation. Bestätigen Sie die Sicherheitsabfragen alle mit 'ja'

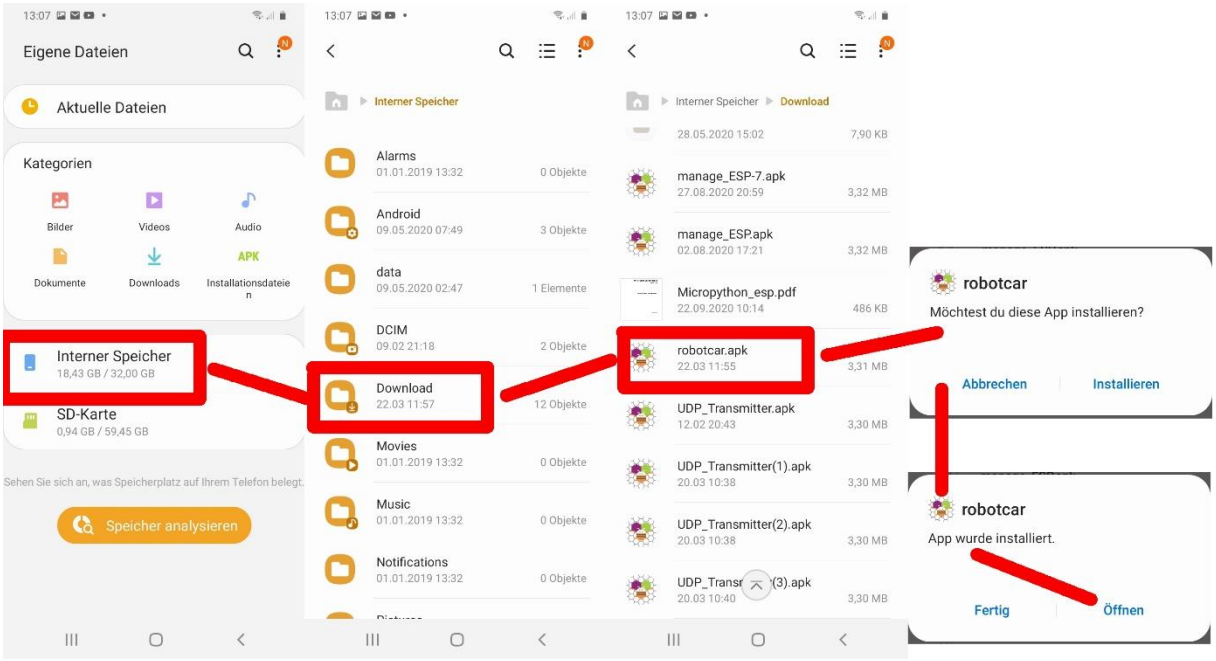


Die zweite Option bietet nach der Übersetzung den Download auf Ihren PC an. Speichern Sie die apk-Datei in einem beliebigen Verzeichnis und transferieren Sie sie danach mit Bluetooth oder einer anderen Methode (e-Mail, WWW, …) auf Ihr Handy. robotcar.apk (apk = **A**ndroid **P**ac**K**age) landet vermutlich im Downloadordner und der liegt in 'Eigene Dateien'. Prüfen Sie zunächst, ob Setups aus diesem Ordner durchgeführt werden dürfen.

Einstellungen …



Eigene Dateien …



Nach dem finalen Test sollten Sie die Erlaubnis, dass Apps aus dem Ordner 'Eigene Dateien' installiert werden dürfen, aus Sicherheitsgründen widerrufen.

Eine Übersicht der Programmblöcke können Sie als PDF herunterladen. Auch die Projektdatei robotcar.aia ist als Download verfügbar, ebenso die Setupdatei robotcar.apk. Ein Listing gibt es für dieses Projekt leider nicht.

Ich wünsche Ihnen viel Freude am weiteren Experimentieren mit Micropython und dem Robot Car. Erweitern sie doch die Handysteuerung mit der Freigabe des Starts für das autonome Fahren Ihres e-Autos. Natürlich müssen dafür noch Routinen, zum Beispiel für das Folgen einer Spurlinie, geschrieben werden. Mit dem Hinzufügen eines weiteren Tasters zur ESP32-Handsteuerung lässt sich das Gleiche dort auch umsetzen. Oder wie wäre es mit einer automatischen Garagentorsteuerung via Infrarot-Modulen bei Annäherung des Autos? Der ESP32 bietet dazu alle Voraussetzungen.

Catch the challenge!

Linkliste:

Teil1	HTML-Format
Teil2	HTML-Format
Teil1	deutsches PDF
	englisches PDF
Teil2	deutsches PDF
	englisches PDF
Teil3	deutsches PDF
	englisches PDF