MAKEKIT

Air:bit

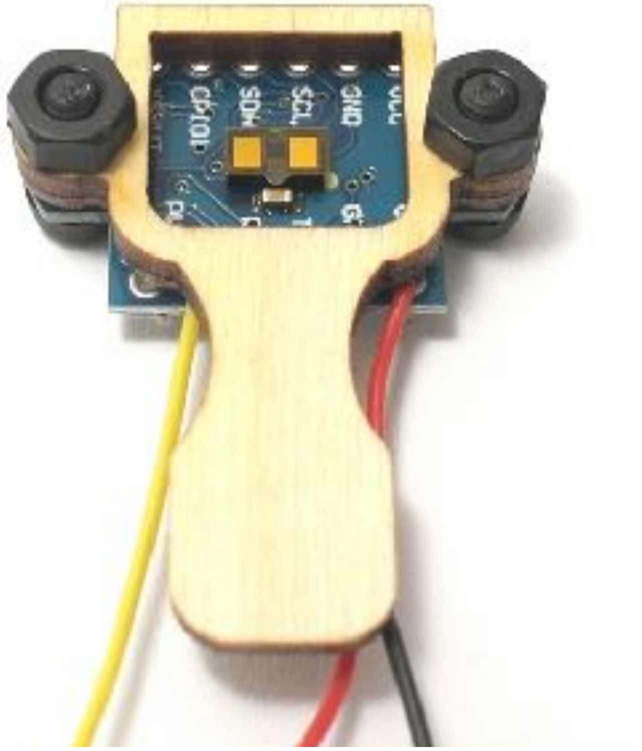
Sensor kit

Lidar

Key specs:

GY-53 Infrared rangefinder

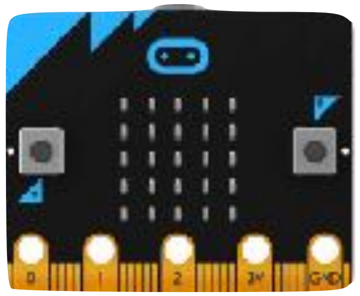
Working principle:	Infrared laser time of flight rangefinder
Measuring distance:	0-2 meter
Operating voltage:	3-5V
Frequency:	20 Hz
Power consumption:	25 mAh
Working temperature:	-20 - 85°C
Weight:	gram



Mount the lidar

Tools: 5,5mm pipe wrench

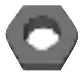
Parts:



1x micro:bit
from drone

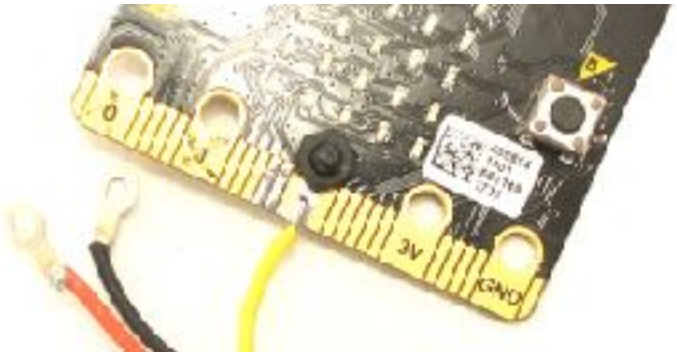
1x Lidar

3x
m3x8mm

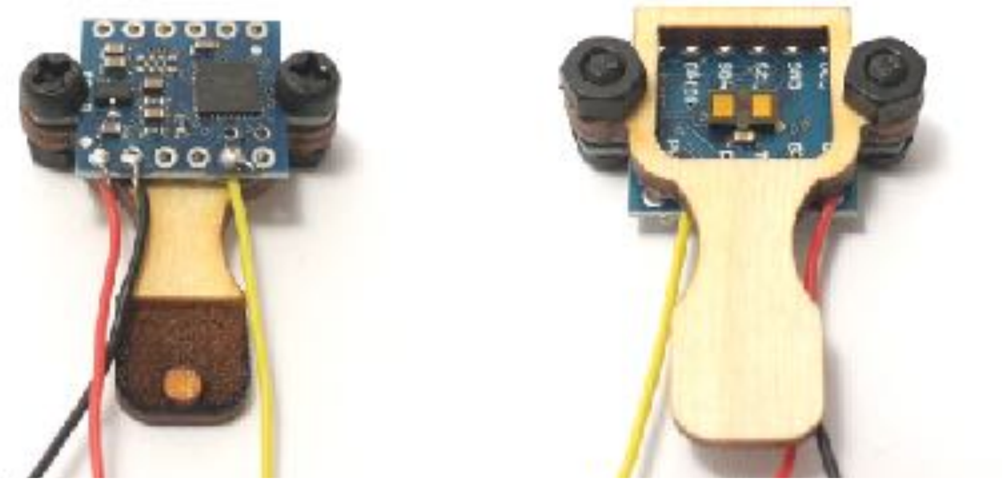


3x
m3 nuts

1x Lidar
holder



Unscrew / disconnect microbit from the drone
With the screw and the nut, screw the yellow cable´s eyelet to P2 on the microbit.
Make sure the eyelet doesn´t connect to the nearby connectors

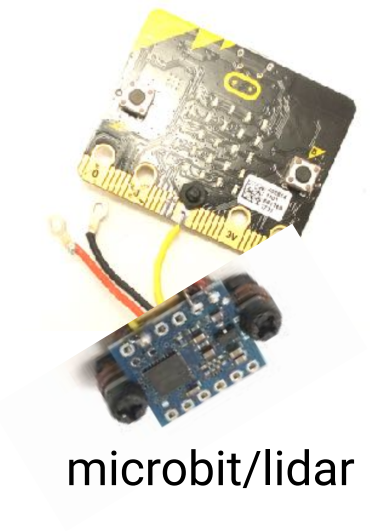


With 2 screws and nuts, carefully screw the lidar to the lidar holder.

Mount the lidar

Tools: 5,5mm pipe wrench

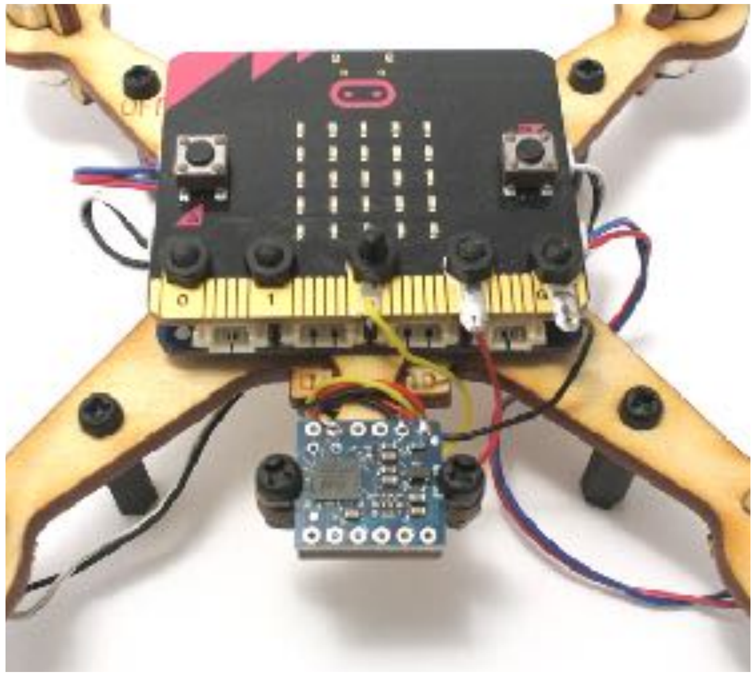
Parts:



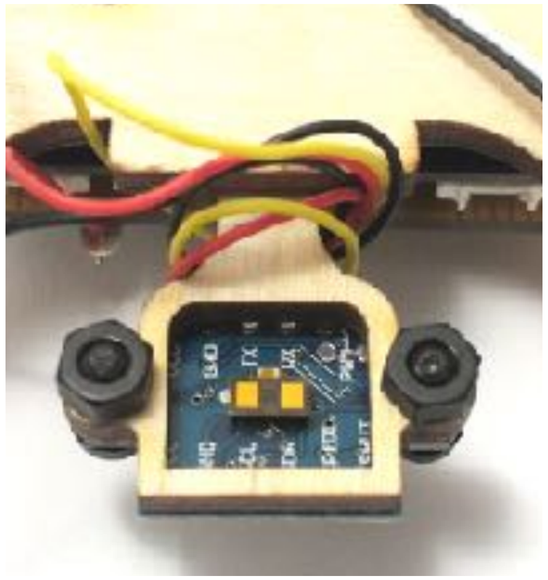
microbit/lidar

Rest of the drone

Left over nuts from Air:bit



Backside



Connect the red cable to 3V and the black cable to GND.
Screw the four nuts back on.
Click the holder in place

Notice! Keep the lidar cables away from the propellers, or they can be cut.

Sonar

Key specs:

US-42 Sonar

Working principle:	Ultrasonic sound
Operating voltage:	3-5V
Measuring distance:	20 cm-7,2 m (5V)
Frequency:	15 Hz
Power consumption:	9 mAh (5V)
Working temperature:	-20 - 65°C
Weight:	5 gram

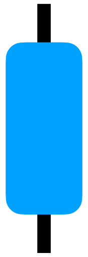
Connect the cables

Tools: Soldering equipment

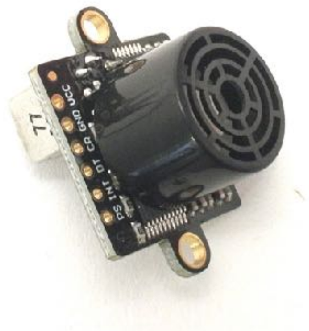
Parts:

Eyelets

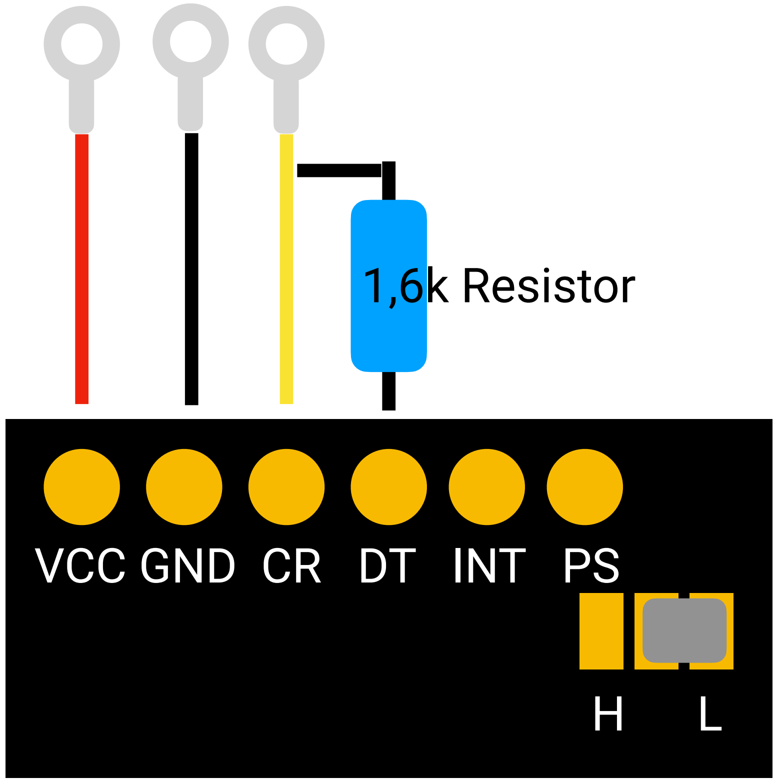
Cables

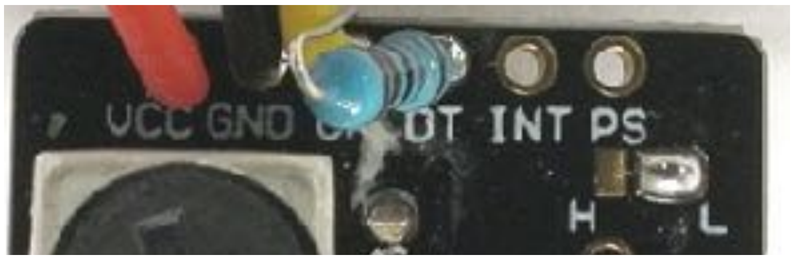


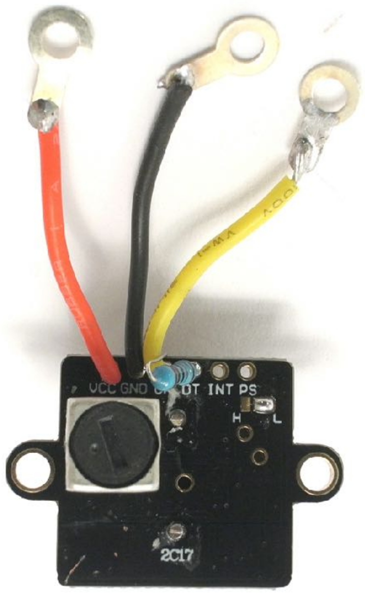
1,6K resistor



Sonar





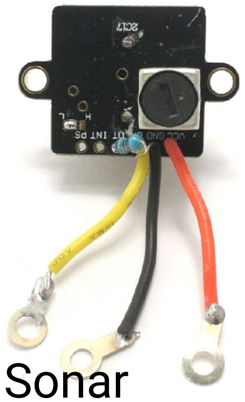


Solder eyelets in one end of each cable
Solder a 1,6k resistor between DT and CR
Solder the yellow signal cable to CR
Solder VCC and GND power cables
Short L and center with some tin to enable PWM output

Mount the sonar

Tools: Soldering equipment

Parts:

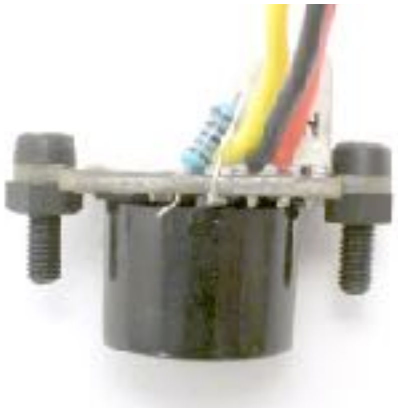


Sonar

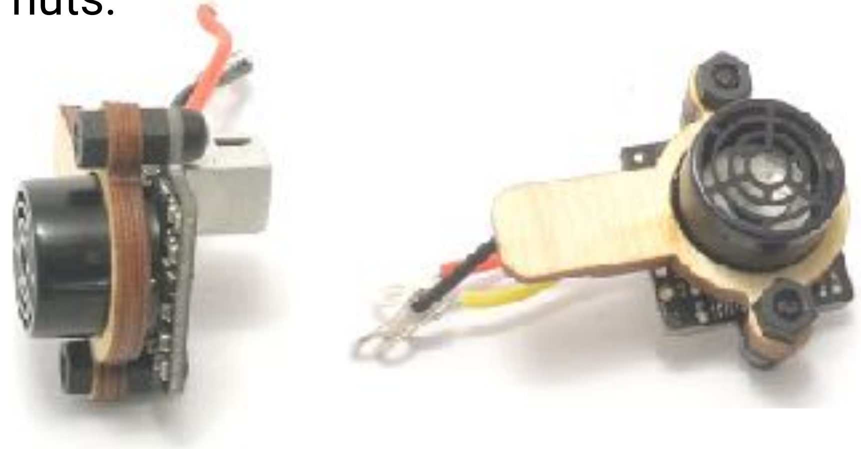
1x
Sonar holder

2x
m3x10mm

4x
m3 nuts

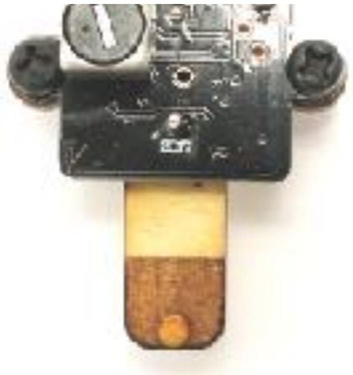


Mount the screw into the sonar holes and add two nuts.



Insert the sonar holder and fasten the 2 last nuts.

Notice that the dark groove is on the backside.

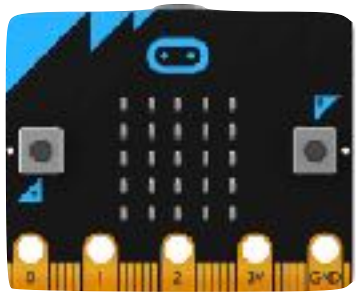


Mount the sonar

Tools: Soldering equipment

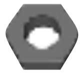
Parts:

Sonar

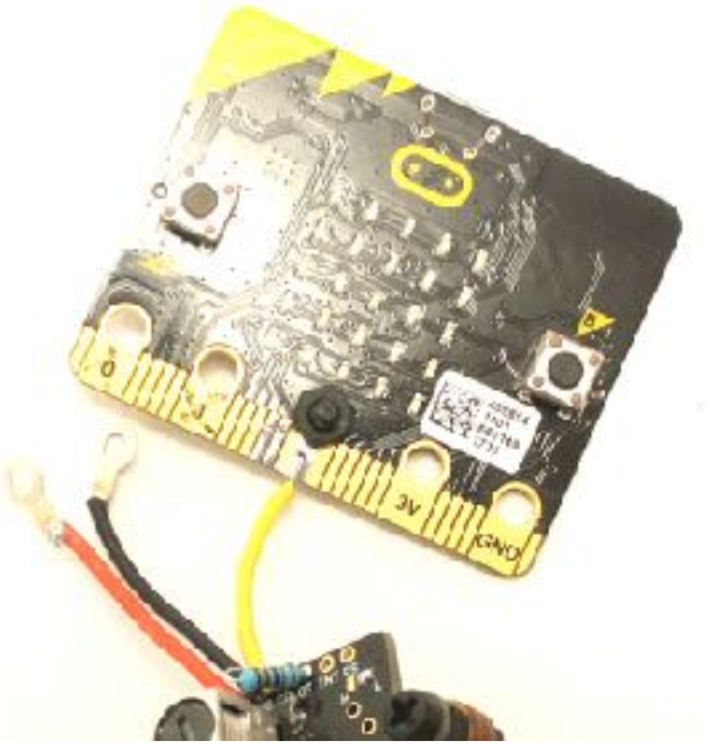


1x micro:bit
from drone

1x
m3x8mm



1x
m3 nut



Unscrew / disconnect microbit from the drone
With the screw and the nut, screw the yellow cable´s eyelet to P2 on the microbit
Make sure the eyelet doesn´t connect to the nearby connectors

Final mounting

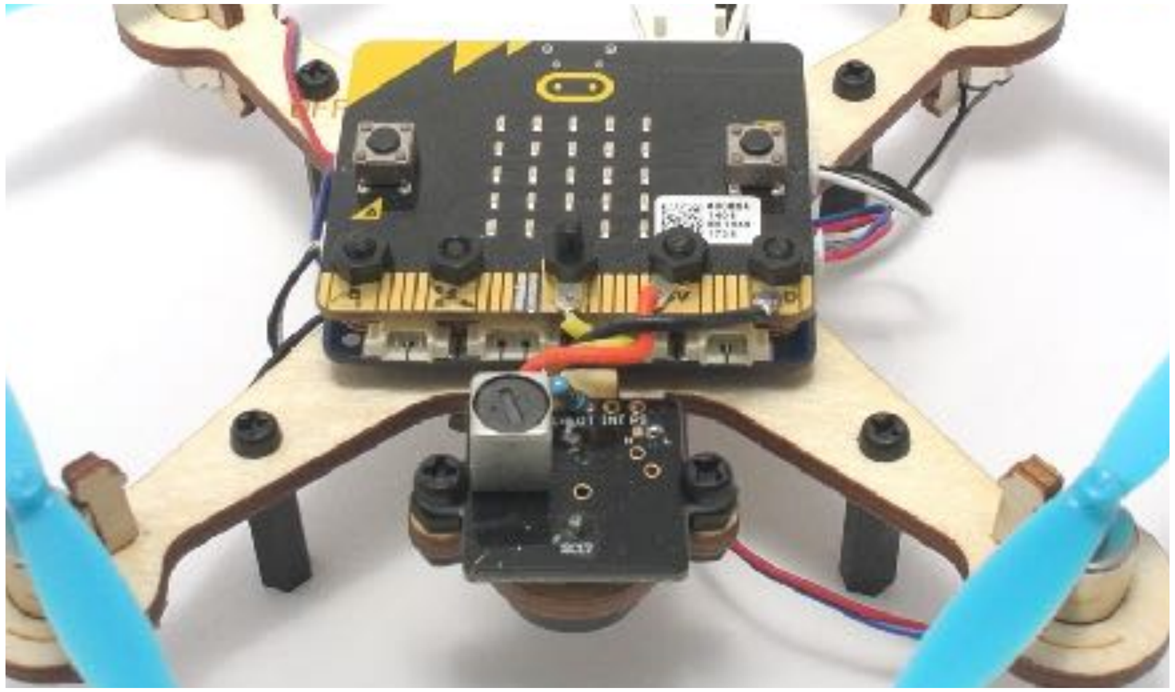
Tools: 5,5mm pipe wrench

Parts:

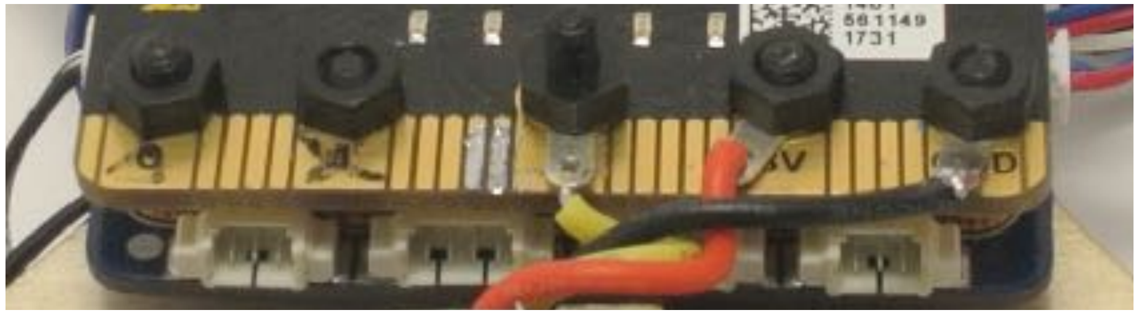
microbit/sonar

Rest of the drone

Left over nuts from Air:bit



Make sure power is off.
Connect the black cable to GND on microbit
Connect the red cable to 3V on microbit
Insert and thighten the remaining 4 nuts
Click the sonar holder on place right under P2



Code it

Download the “microbit-Airbit-drone-gr-7.hex” from air:bit support page
There is a separate readymade “cheatcode” similar to this tutorial called “microbit-Airbit-Drone-Sonar.hex”

Altitude hold

Download the “code for drone” from air:bit support page
Create 4 new variables for our altitude hold function:

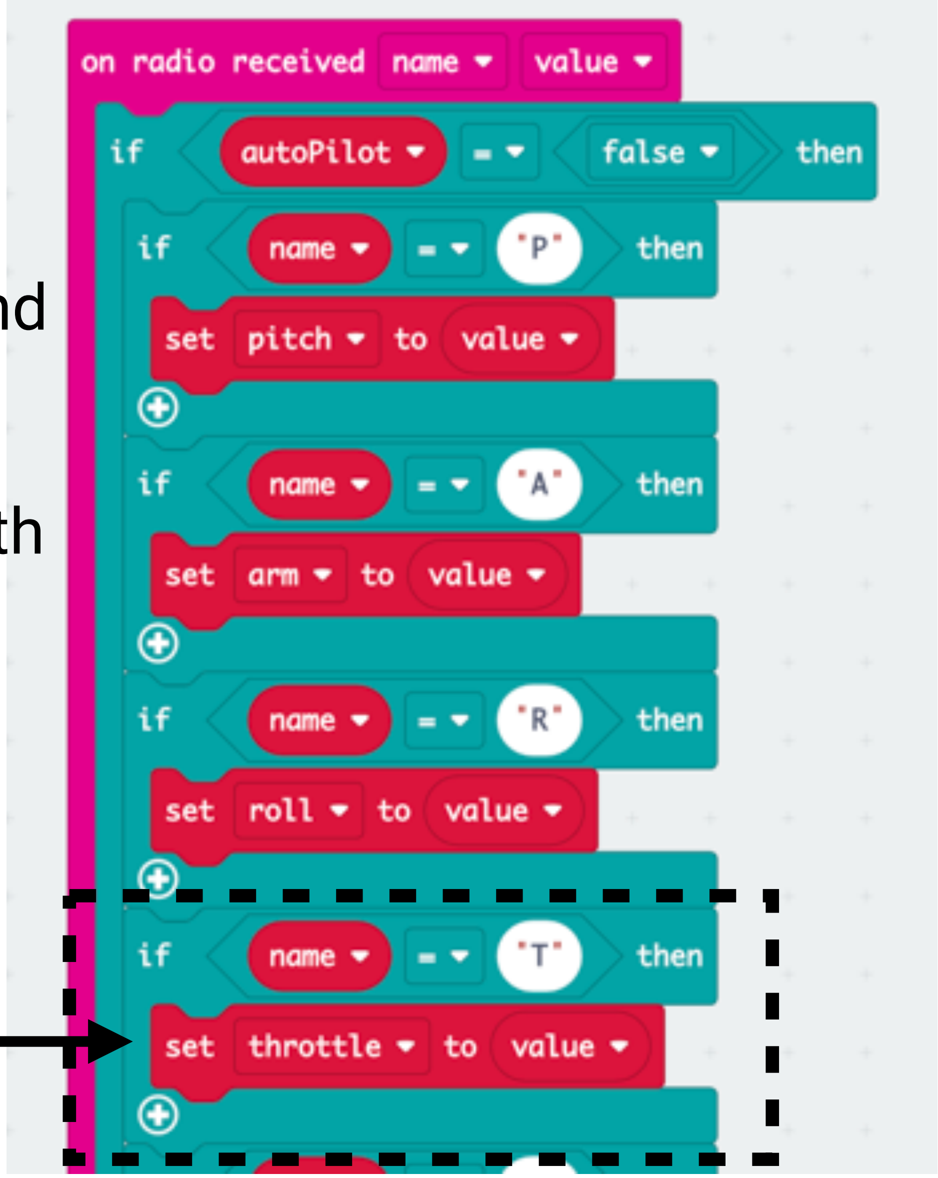
New variables:

desiredAltitude	The altitude (in cm) we want drone to be
throttleP	The compensation force we add to throttle (Proportional to the error)
throttleMid	The approximate valute where the drone will stay at an even height
AltitudeError	The gap between actual altitude and desired altitude (in cm)

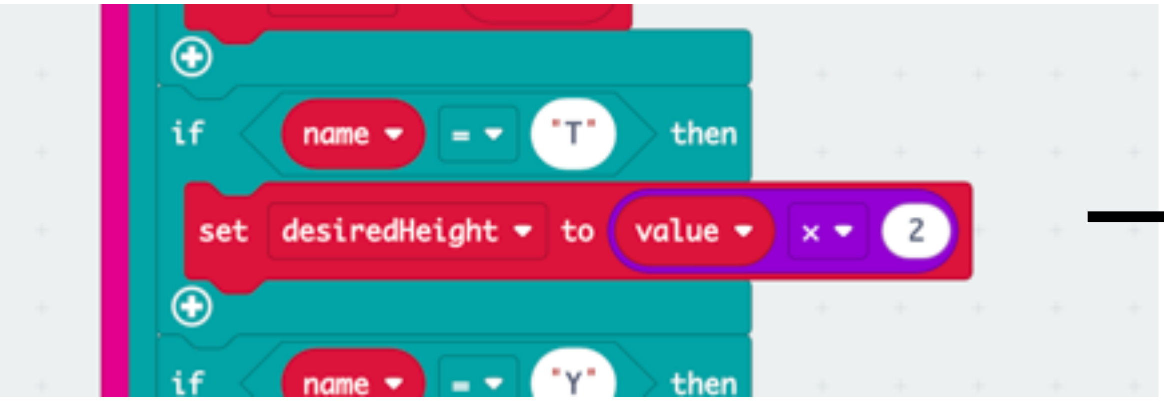
Get the desired altitude

Throttle is normally a number between 0 and 100
We will convert this to desired altitude (cm)
As a start, let´s make an altitude range between 0 and 2m

In the on radio received, replace the throttle entry with desiredHeight * 2.
Desiredheight will now contain our altitude in cm.



```
on radio received name value
  if autoPilot = false then
    if name = "P" then
      set pitch to value
    if name = "A" then
      set arm to value
    if name = "R" then
      set roll to value
    if name = "T" then
      set throttle to value
```



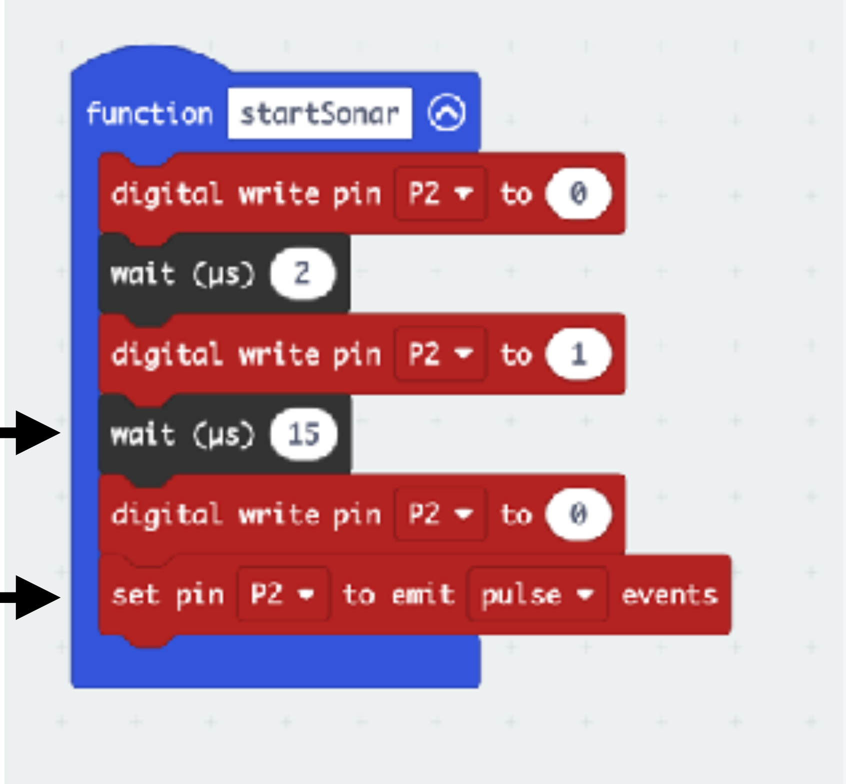
```
if name = "T" then
  set desiredHeight to value × 2
if name = "Y" then
```

Send out the sonar echo

Make a function for starting the sonar, then wait for the pulse back from sensor

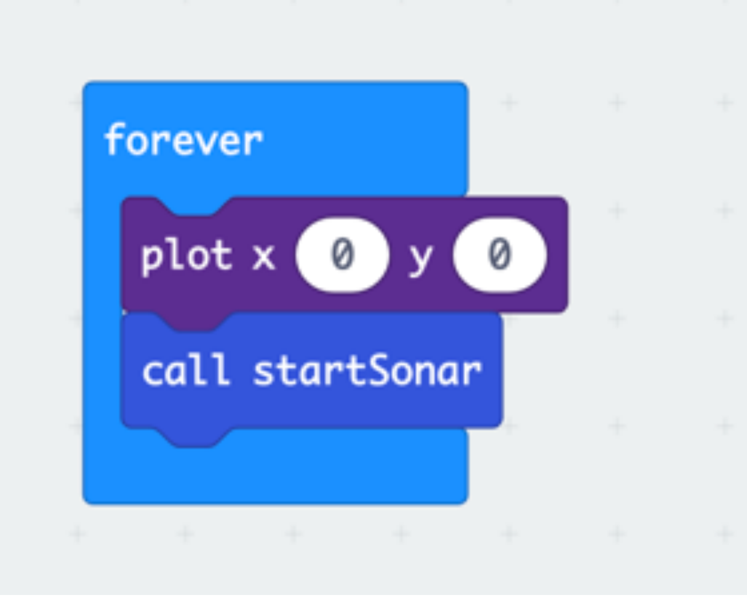
Setting P2 hight for at least 10 microseconds will make the sonar send out an echo

Setup pin 2 to receive pulse events back from sensor



```
function startSonar
    digital write pin P2 to 0
    wait (µs) 2
    digital write pin P2 to 1
    wait (µs) 15
    digital write pin P2 to 0
    set pin P2 to emit pulse events
```

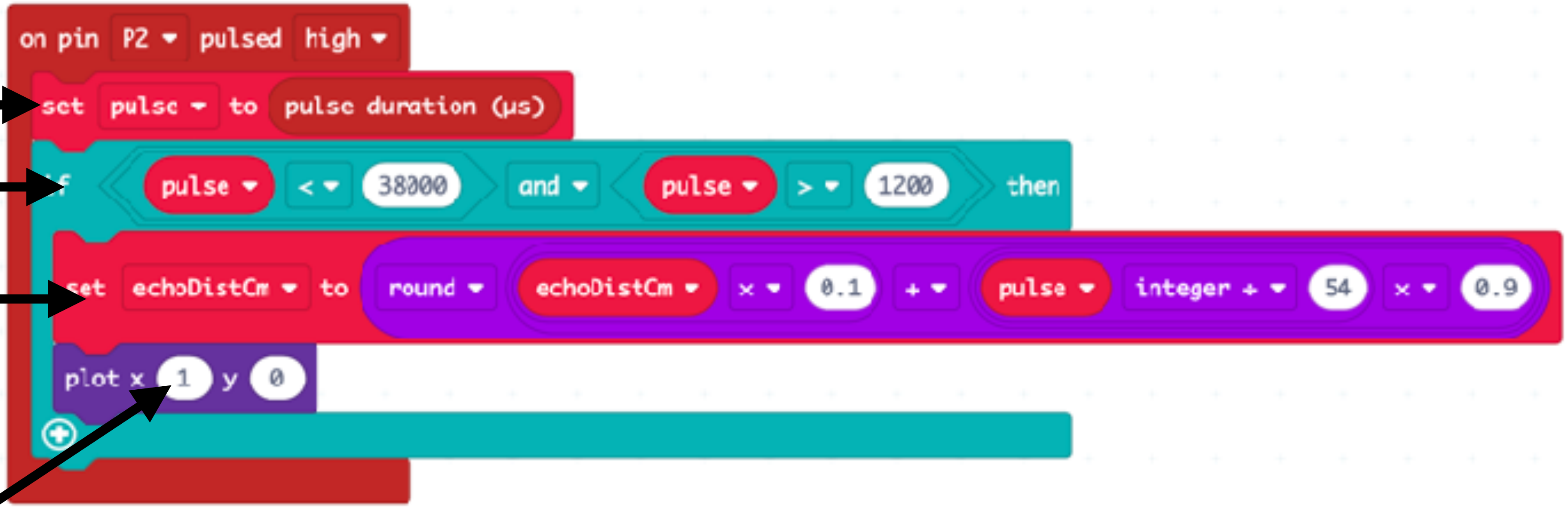
In a separate forever loop, run the function. The plot 0,0 tells you that a sonar sound is being fired by blinking the top left LED on the screen.



```
forever
    plot x 0 y 0
    call startSonar
```

Receive and measure echo distance

Make an interrupt that activates everytime there is a received signal pulse from sonar.



```
on pin P2 ▾ pulsed high ▾
    set pulse ▾ to pulse duration (µs)
    if pulse ▾ < ▾ 38000 and ▾ pulse ▾ > ▾ 1200 then
        set echoDistCm ▾ to round ▾ echoDistCm ▾ × ▾ 0.1 + ▾ pulse ▾ integer ÷ ▾ 54 × ▾ 0.9
        plot x 1 y 0
```

How long did the sound travel to object and back?

Check if result is ok (within normal range)

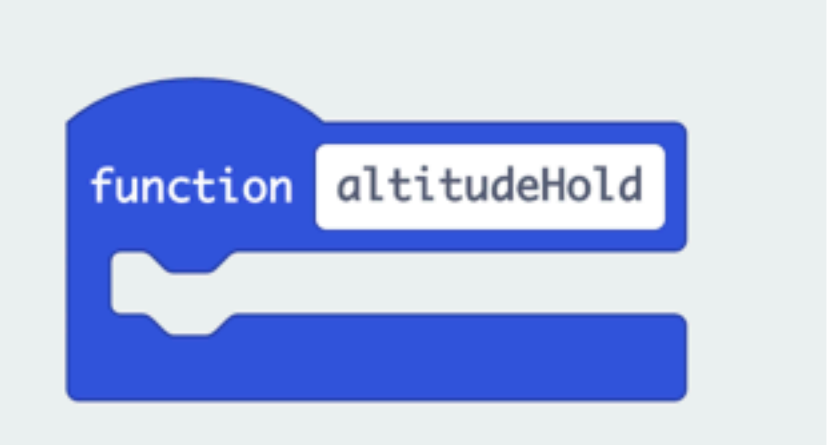
This formula converts from the sound travel time to cm And smooths the result

Plot a led when received successfull echo, will let you know everything is working

Read more about sonar and distance:
https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/

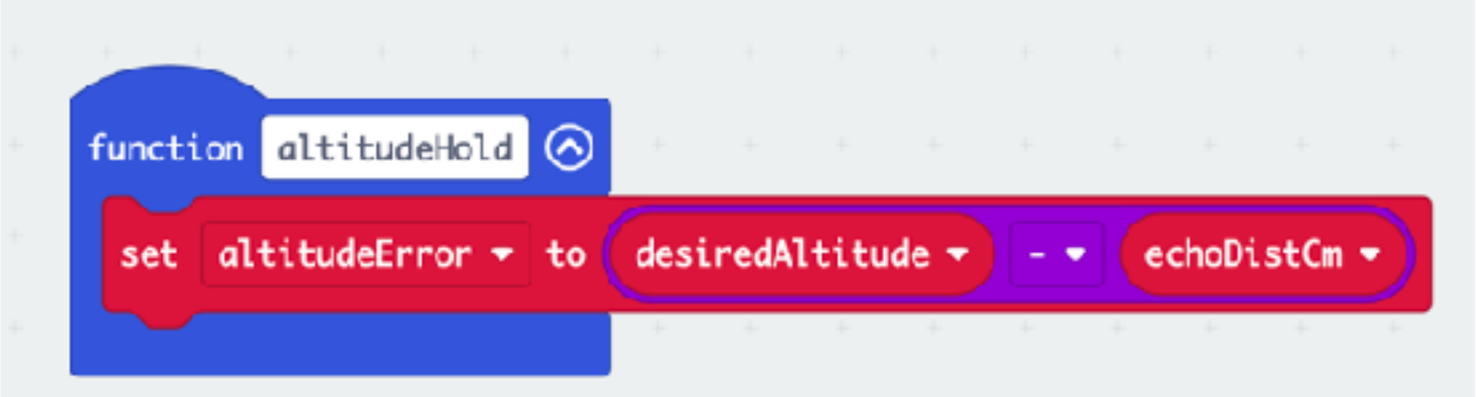
Make the altitude hold function

Press function - create function called altitudeHold



```
function altitudeHold
```

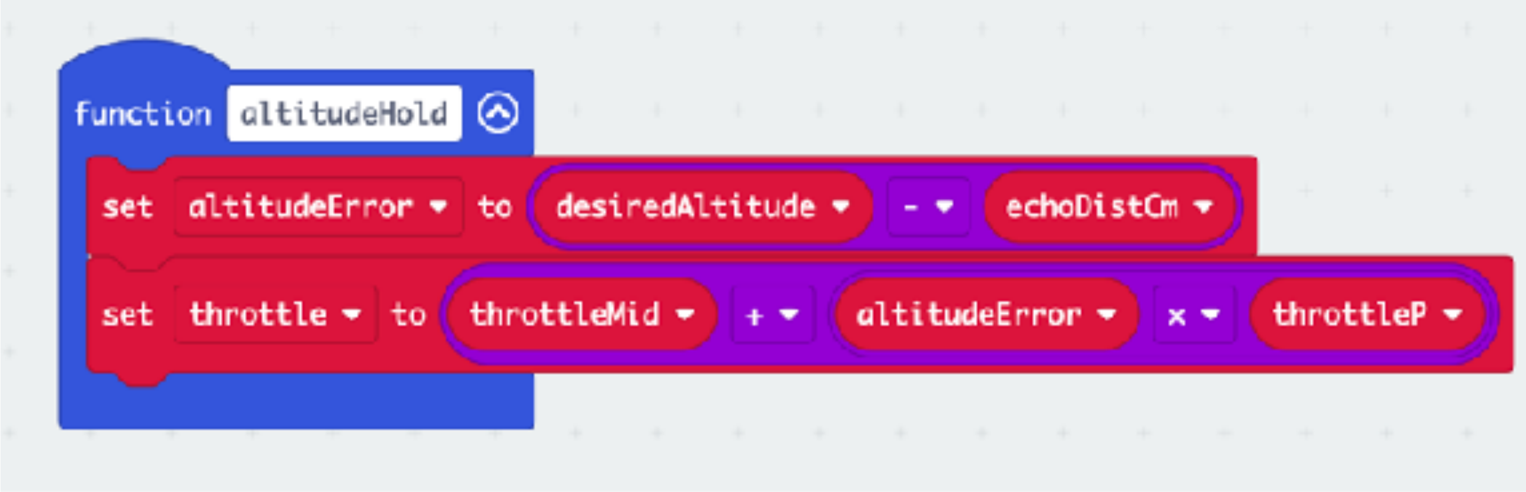
Set altitudeError to desiredAltitude - echoDistCm To get the gap between actual height and desired height



```
function altitudeHold
    set altitudeError to desiredAltitude - echoDistCm
```

Set throttle to the medium throttle + altitudeError * our proportional power.

This will add a certain change in throttle to climb or ascend



```
function altitudeHold
    set altitudeError to desiredAltitude - echoDistCm
    set throttle to throttleMid + altitudeError x throttleP
```

The proportional power

Modern stabilisation systems uses PID-control (Proportional, Integral and Derivative)

For our altitude, Proportional will work fine by itself. As of writing, we will stick with the P-value (throttleP)

If this value is too low, we will not add enought throttle to climb or climb very slowly.

If value is too high, we will overshoot, and create occillations.

The key is to find a perfect value for P.

0.2 is a good starting point!

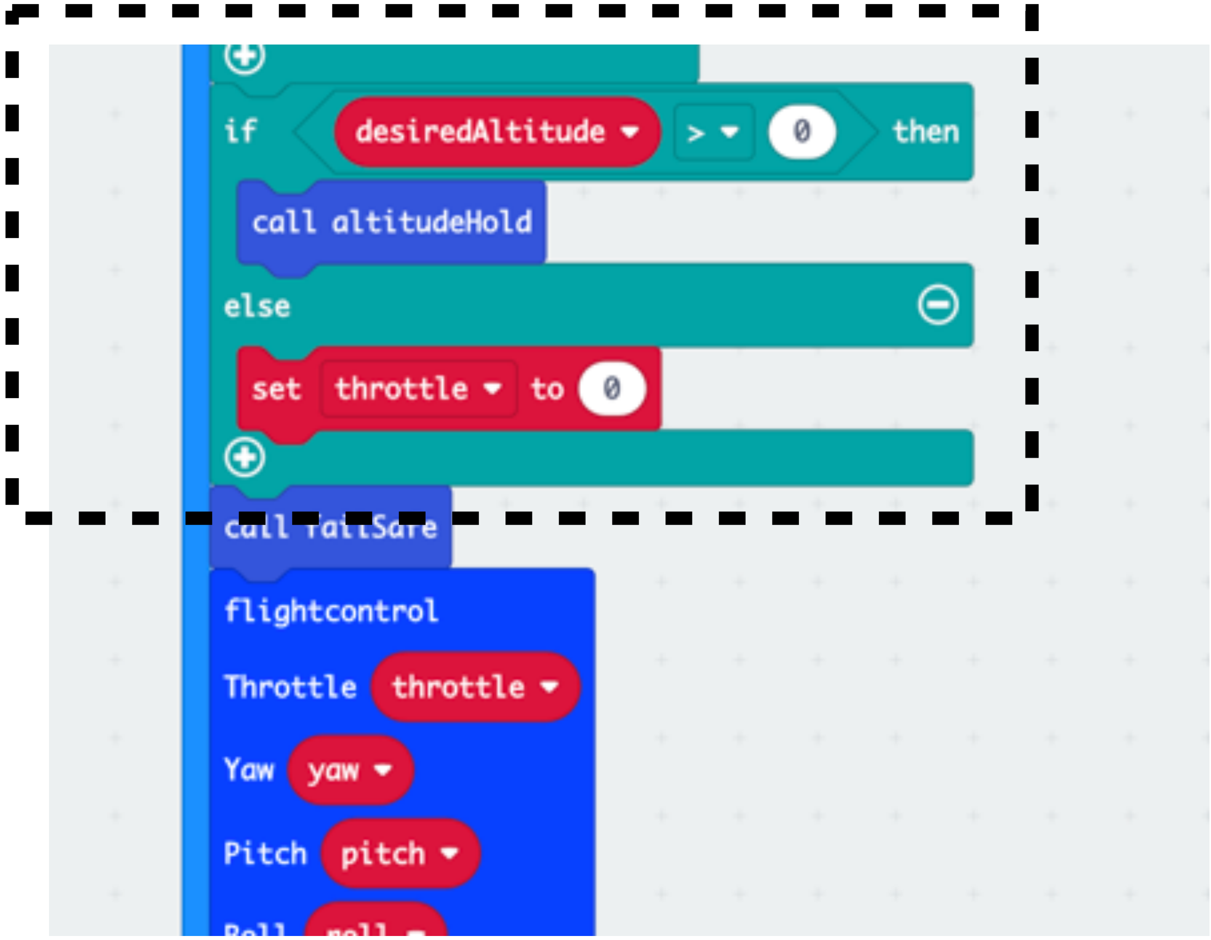
Use the function

In the forever-loop, right above “call failSafe”, insert the blocks shown in the frame.

This will activate the altitude hold when we start increasing the altitude (Pressing button B on our transmitter)

We want throttle to be 0 when we start our drone. This code will make sure throttle stays 0 until we have armed and increased the throttle on the transmitter.

Throttle from transmitter means altitude on our drone.



```
if desiredAltitude > 0 then
    call altitudeHold
else
    set throttle to 0
call failSafe
flightcontrol
Throttle throttle
Yaw yaw
Pitch pitch
Roll roll
```

Test it

Prepare to fly like normal. Arm the motors with A+B. Increase the throttle until drone lifts up.

One click on B increases the height by 10 cm,
One click on A decresases with 10 cm.

Click slowly 10 times on B and the drone climbs to 1 meter then watch the drone keep the height automatically.

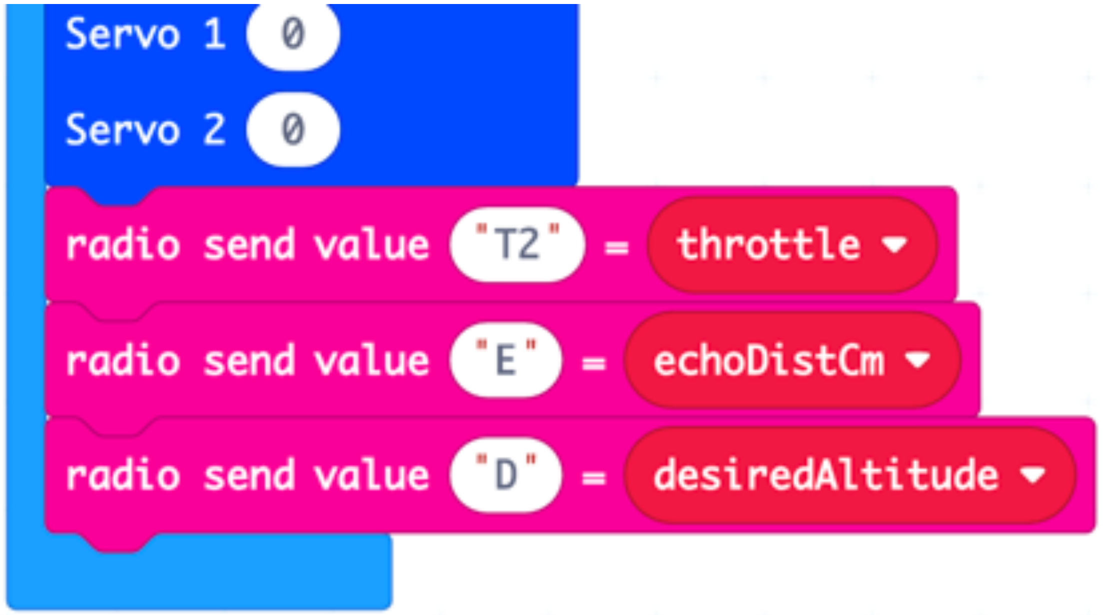
Video:
https://youtu.be/nyt7U8_Gbak

Debug and monitor

If it doesn´t work, you can monitor your values using telemetry.

In the Bottom of drone´s forever loop, add the following:

Drone code:



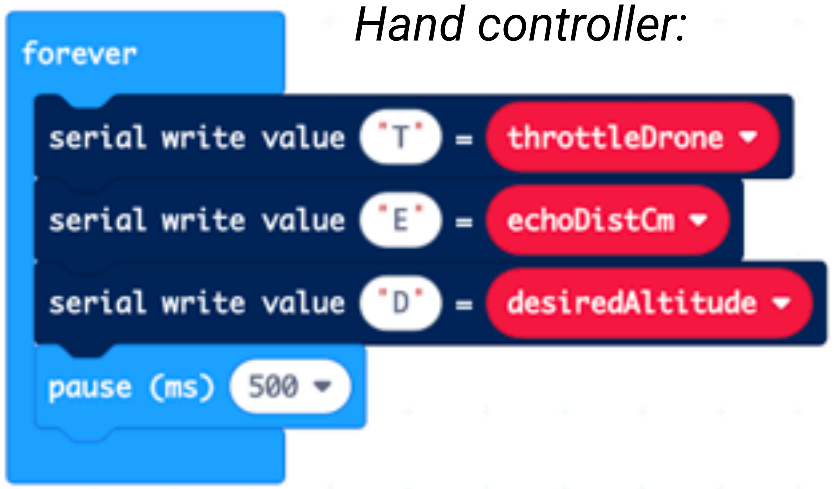
```
Servo 1 0
Servo 2 0
radio send value "T2" = throttle
radio send value "E" = echoDistCm
radio send value "D" = desiredAltitude
```

On your hand controller, add

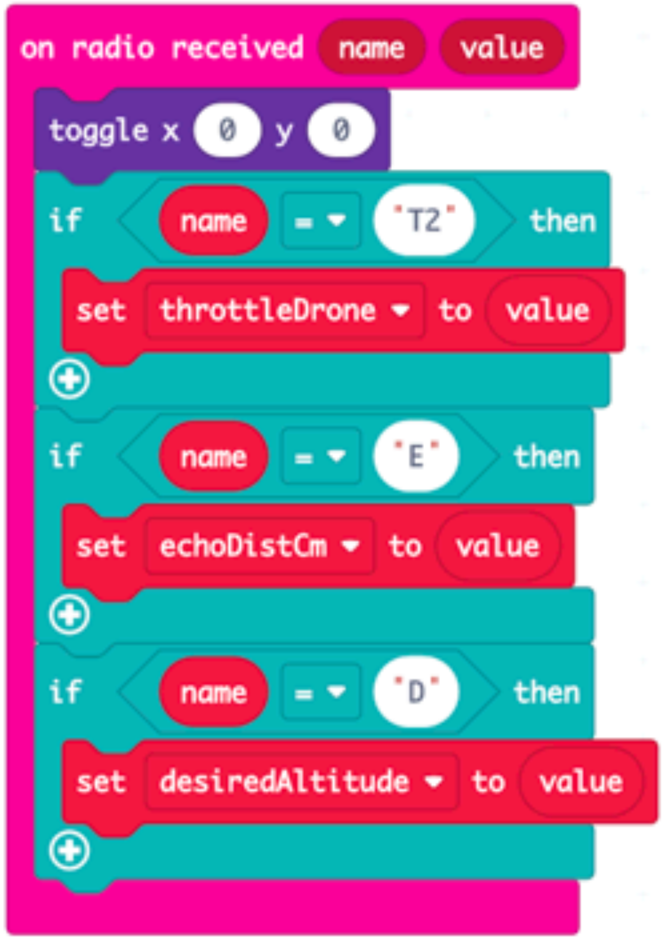
the variables : throttleDrone, EchoDistCm, desiredaltitude

Create a new forever loop, and a radio received event, Just like the blocks on the right

Hand controller:

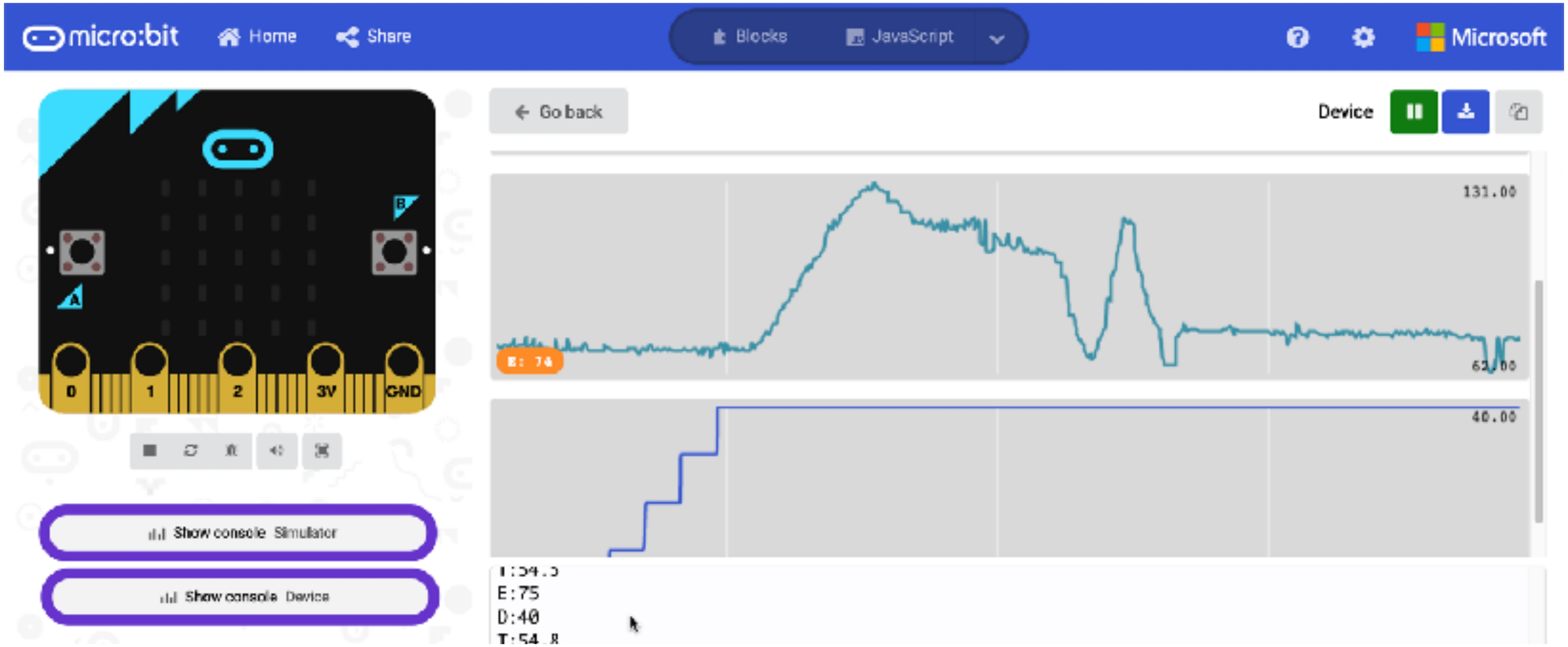


```
forever
    serial write value "T" = throttleDrone
    serial write value "E" = echoDistCm
    serial write value "D" = desiredAltitude
    pause (ms) 500
```



```
on radio received name value
    toggle x 0 y 0
    if name = "T2" then
        set throttleDrone to value
    if name = "E" then
        set echoDistCm to value
    if name = "D" then
        set desiredAltitude to value
```

Monitor it



```
T:54.5
E:75
D:40
T:54.8
```

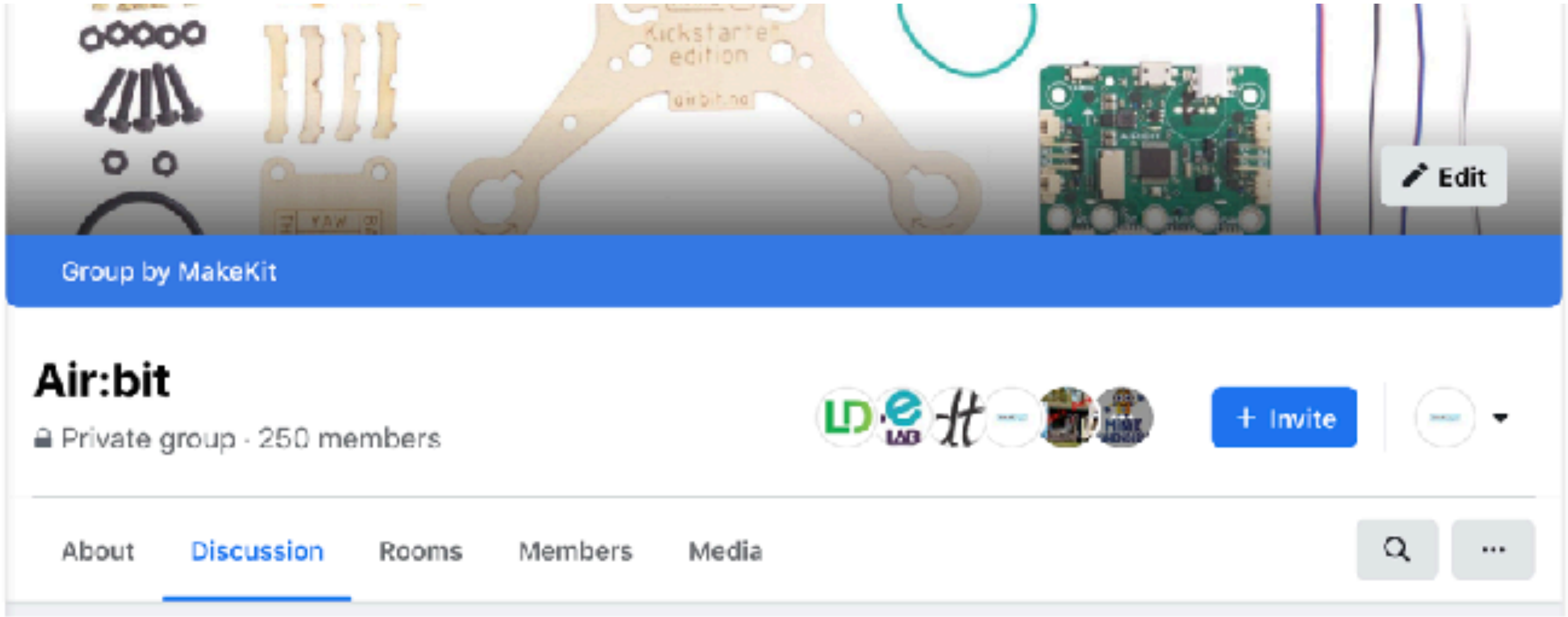
- Keep your hand controller connected to the computer.
- Make sure the microbit is paired and has the latest firmware (details.txt on microbit, Interface Version: 0253)
- Download the code
- Click “Show console device”. (This doesn´t always show up, try downloading again, pairing again, close other tabs in the browser)

Watch the values:

- The E is the measured distance. Watch if it is measuring correctly by holding the drone above ground and move it up and down.
- The D is the desired distance, 0-200 cm as being selected from the remote A or B.
- The T is the throttle automatically being applied to the drone´s motor speed.

Share and discuss:

Get tips and help in our Facebook community:
www.facebook.com/groups/goairbit/



Group by MakeKit

Air:bit

Private group · 250 members

www.makekit.no   support@makekit.no   makekit   gomakekit (also twitter)