**Drone Controlling Program**

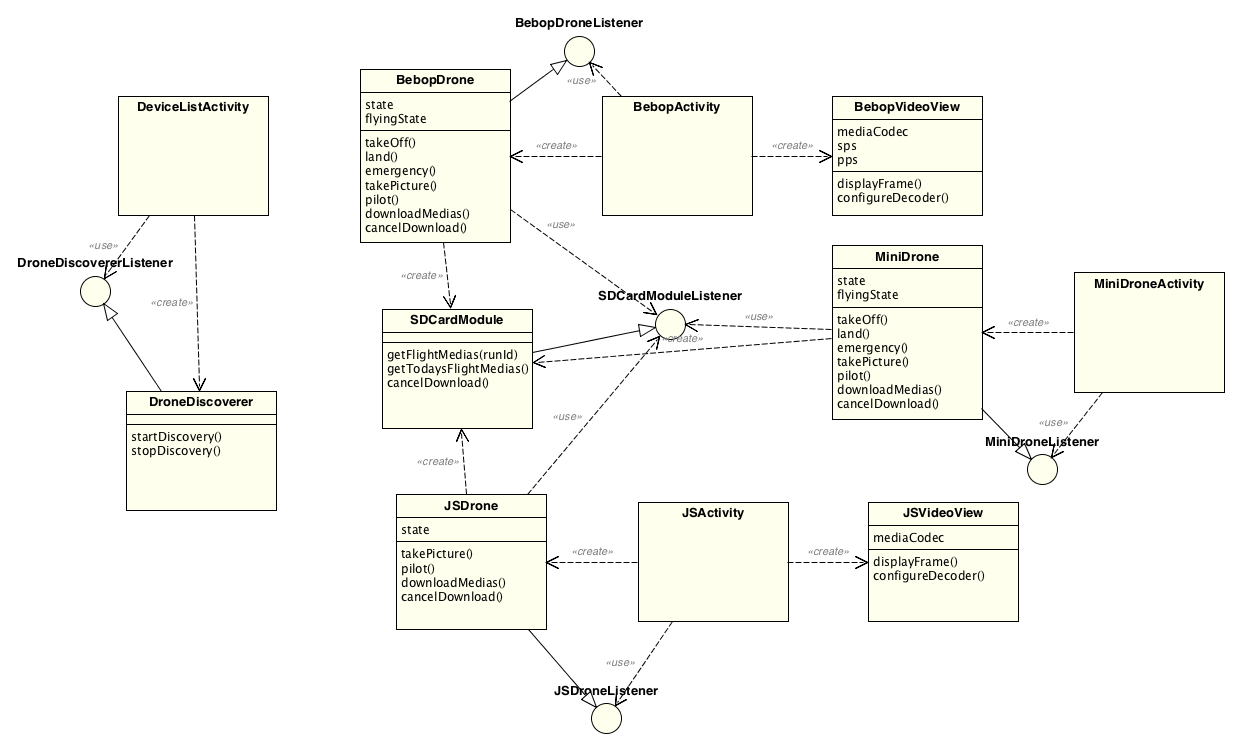
They are standalone, this means that you can clone this repo and use them without compiling the SDK. To enable this, they will use the precompiled SDK libraries.

The mobile samples show you how to connect, pilot, take pictures, display video stream if available, and download medias from the drone. Each mobile sample can be used without having to build the SDK: it will use the precompiled libraries. But you can also use the sample with your own compiled SDK.

**Write code**

You can write some code for your smartphone, your smart watch, or your VR glasses!

The only limitation is your imagination

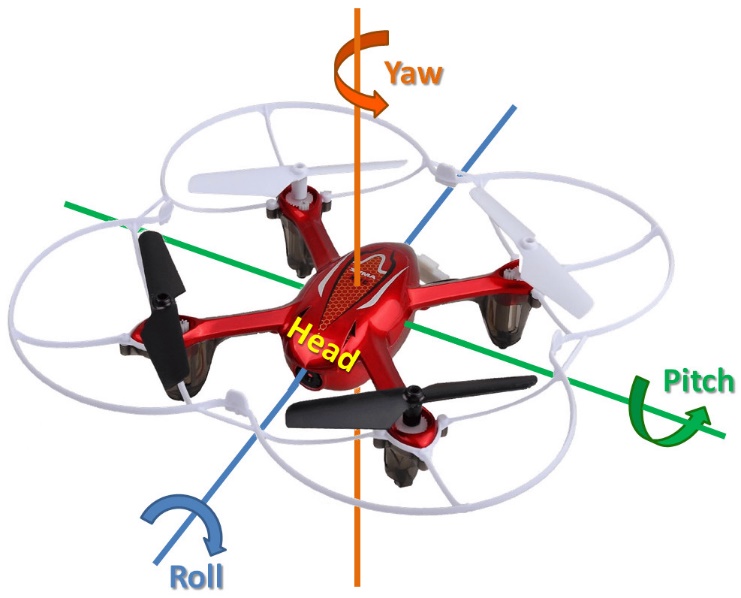


**SDK parrot repository link:**

<https://github.com/Parrot-Developers/Samples>

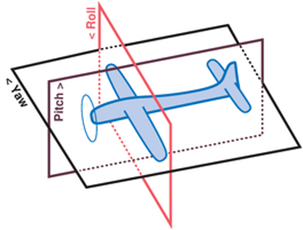
**THE AXIS**

Learn the terminology and underlying mechanics of drone flight. Gaz (moving up and down), roll (left and right movement), pitch (forward and backwards movement), yaw (rotation), and other terms are explained and demonstrated.



**Yaw**

Yaw is simply defined as the rotation of a UAV with respect to the center axis. If you were looking down on a drone from above, yaw refers to the movement of the drone clockwise or counterclockwise. In short, if your drone is spinning round-and-round like a Tilt-a-Whirl, you may have a problem (unless you’re into aerial acrobatics, of course) as a yaw spin often precedes a crash with regard to copters.

**Pitch**

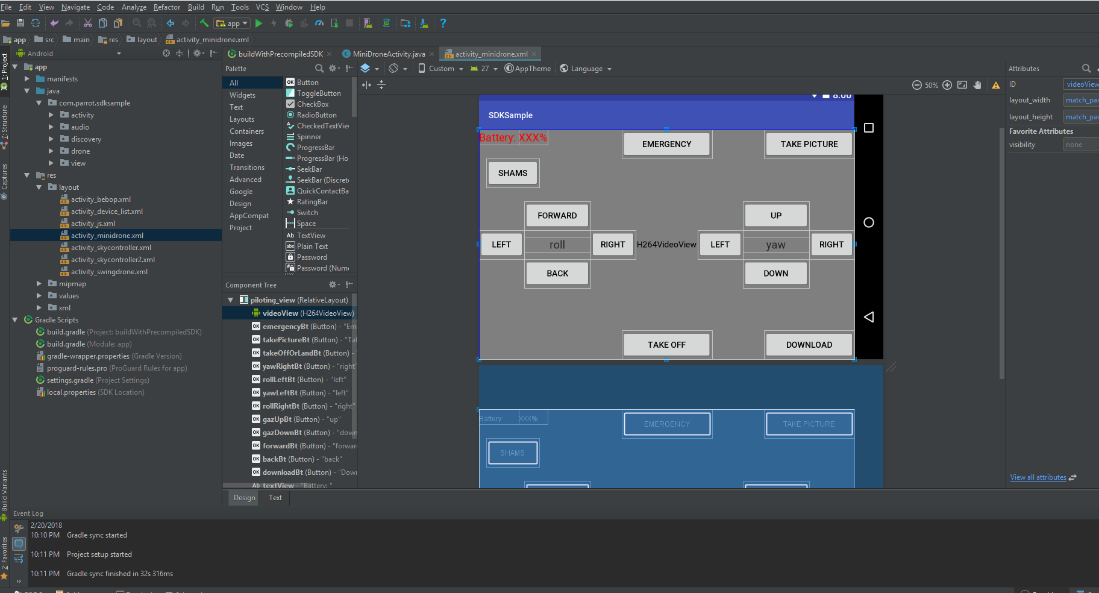
Another aerial flight term, pitch simply describes movement up and down along the vertical axis from the front to the back of the drone (or nose to tail in an airplane config).

**Roll**

The final aerial movement term, roll is exactly what it sounds like, the rotation of the aircraft from nose to tail (front to back on a copter). Basically roll refers to the movement of the drone forward, backward, left and right along a horizontal axis. Unless you’re an advanced, acrobatic UAV pilot (and why would you be reading this article if you were?), you will want to keep yaw, pitch and roll in as stable a position as possible.

**Drone controlling**

1. Take off
2. Land
3. Emergency
4. Forward
5. Back
6. Right
7. Left
8. Up
9. Down
10. Turn Right
11. Turn Left



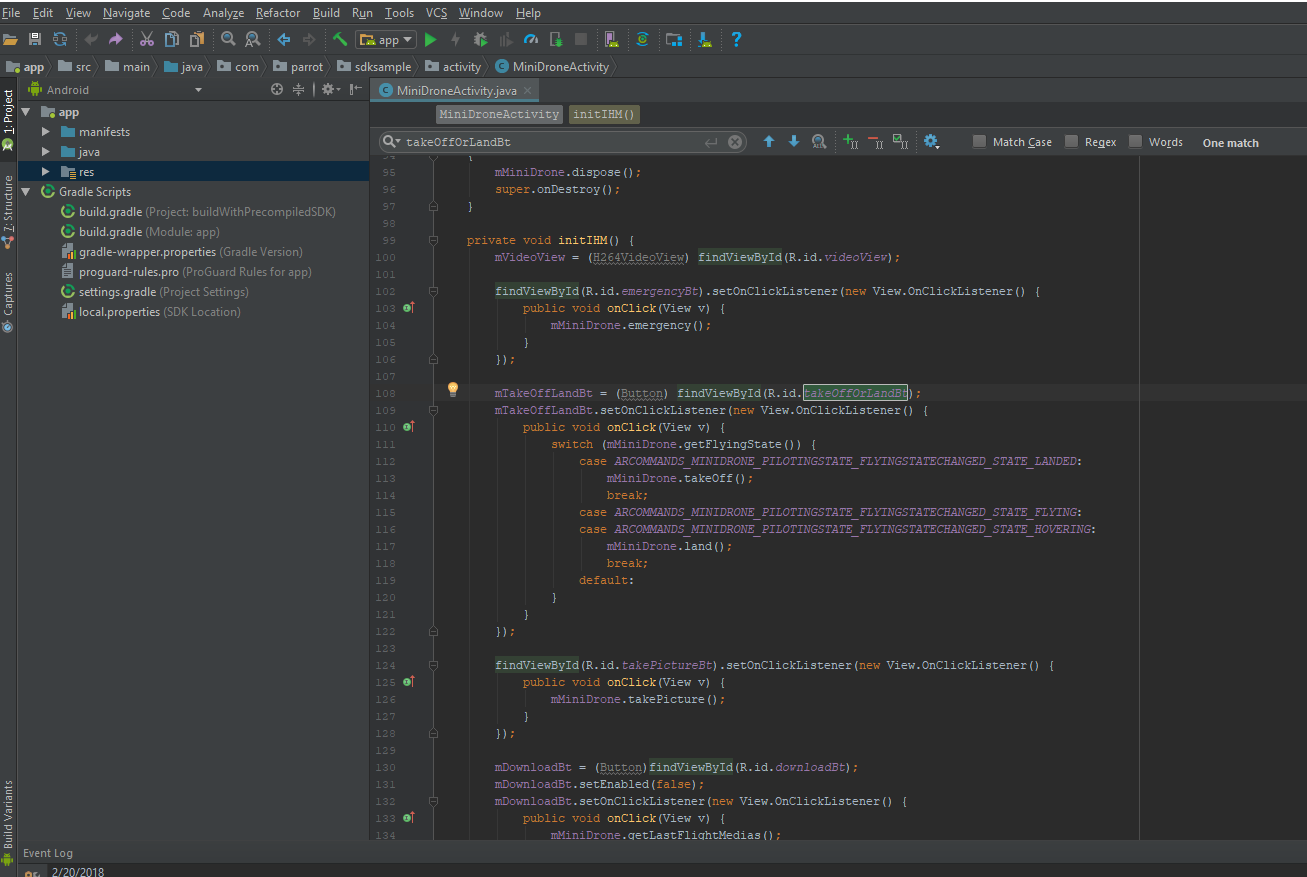
**TAKING OFF**

In order to make your drone take off you will need to ensure that its flying status is landed (even if you can send take off commands anyway, it just won’t take of if it not in landed state).

Then, you can send the takeoff command.

In response, your drone will send you a state change (if it has taken off): State Landed -> State TakingOff -> State Hovering (or Flying).

After that, you can start piloting your drone.



**LANDING**

When you’re done flying, you will need to land. This is how you do it: simply send the landing command which will land the drone.

mTakeOffLandBt = (Button) findViewById(R.id.*takeOffOrLandBt*);  
mTakeOffLandBt.setOnClickListener(new View.OnClickListener() {  
 public void onClick(View v) {  
 switch (mMiniDrone.getFlyingState()) {  
 case *ARCOMMANDS\_MINIDRONE\_PILOTINGSTATE\_FLYINGSTATECHANGED\_STATE\_LANDED*:  
 mMiniDrone.takeOff();  
 break;  
 case *ARCOMMANDS\_MINIDRONE\_PILOTINGSTATE\_FLYINGSTATECHANGED\_STATE\_FLYING*:  
 case *ARCOMMANDS\_MINIDRONE\_PILOTINGSTATE\_FLYINGSTATECHANGED\_STATE\_HOVERING*:  
 mMiniDrone.land();  
 break;  
 default:  
 }  
 }  
});

**EMERGENCY**

When you need to stop the drone immediately, you send emergency command. It will turn off immediately not like the landing.

findViewById(R.id.*emergencyBt*).setOnClickListener(new View.OnClickListener() {  
 public void onClick(View v) {  
 mMiniDrone.emergency();  
 }  
});

**FORWARD**

When you want to move the drone forward you need to push on the forward button and once you release the button the drone return to its original parameters.

findViewById(R.id.*forwardBt*).setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
 v.setPressed(true);  
 mMiniDrone.setPitch((byte) 50);  
 mMiniDrone.setFlag((byte) 1);  
 break;  
  
 case MotionEvent.*ACTION\_UP*:  
 v.setPressed(false);  
 mMiniDrone.setPitch((byte) 0);  
 mMiniDrone.setFlag((byte) 0);  
 break;  
  
 default:  
  
 break;  
 }  
  
 return true;  
 }  
});

**BACK**

When you want to move the drone back you need to push on the back button and once you release the button the drone return to its original parameters

findViewById(R.id.*backBt*).setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
 v.setPressed(true);  
 mMiniDrone.setPitch((byte) -50);  
 mMiniDrone.setFlag((byte) 1);  
 break;  
  
 case MotionEvent.*ACTION\_UP*:  
 v.setPressed(false);  
 mMiniDrone.setPitch((byte) 0);  
 mMiniDrone.setFlag((byte) 0);  
 break;  
  
 default:  
  
 break;  
 }  
  
 return true;  
 }  
});

**ROLL LEFT**

findViewById(R.id.*rollLeftBt*).setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
 v.setPressed(true);  
 mMiniDrone.setRoll((byte) -50);  
 mMiniDrone.setFlag((byte) 1);  
 break;  
  
 case MotionEvent.*ACTION\_UP*:  
 v.setPressed(false);  
 mMiniDrone.setRoll((byte) 0);  
 mMiniDrone.setFlag((byte) 0);  
 break;  
  
 default:  
  
 break;  
 }  
  
 return true;  
 }  
});

**ROLL RIGHT**

@Override  
 public boolean onTouch(View v, MotionEvent event) {  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
 v.setPressed(true);  
 mMiniDrone.setRoll((byte) 50);  
 mMiniDrone.setFlag((byte) 1);  
 break;  
  
 case MotionEvent.*ACTION\_UP*:  
 v.setPressed(false);  
 mMiniDrone.setRoll((byte) 0);  
 mMiniDrone.setFlag((byte) 0);  
 break;  
  
 default:  
  
 break;  
 }  
  
 return true;  
 }  
});

**UP**

findViewById(R.id.*gazUpBt*).setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
 v.setPressed(true);  
 mMiniDrone.setGaz((byte) 50);  
 break;  
  
 case MotionEvent.*ACTION\_UP*:  
 v.setPressed(false);  
 mMiniDrone.setGaz((byte) 0);  
 break;  
  
 default:  
  
 break;  
 }  
  
 return true;  
 }  
});

**DOWN**

findViewById(R.id.*gazDownBt*).setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
 v.setPressed(true);  
 mMiniDrone.setGaz((byte) -50);  
 break;  
  
 case MotionEvent.*ACTION\_UP*:  
 v.setPressed(false);  
 mMiniDrone.setGaz((byte) 0);  
 break;  
  
 default:  
  
 break;  
 }  
  
 return true;  
 }  
});

**YAW (ROTATION) LEFT**

findViewById(R.id.*yawLeftBt*).setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
 v.setPressed(true);  
 mMiniDrone.setYaw((byte) -50);  
 break;  
  
 case MotionEvent.*ACTION\_UP*:  
 v.setPressed(false);  
 mMiniDrone.setYaw((byte) 0);  
 break;  
  
 default:  
  
 break;  
 }  
  
 return true;  
 }  
});

**YAW RIGHT**

findViewById(R.id.*yawRightBt*).setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
 v.setPressed(true);  
 mMiniDrone.setYaw((byte) 50);  
 break;  
  
 case MotionEvent.*ACTION\_UP*:  
 v.setPressed(false);  
 mMiniDrone.setYaw((byte) 0);  
 break;  
  
 default:  
  
 break;  
 }  
  
 return true;  
 }  
});

Add Search button

Set name to search\_bt

Add functionality to this button:

Go to Java- miniDroneActivity- add the below code to do the following:

1. SetPitch (forward 1 sec)

findViewById(R.id.*search\_bt*).setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 try {  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
  
 mMiniDrone.setPitch((byte) 50);  
 mMiniDrone.setFlag((byte) 1);  
 Thread.*sleep*(1000);  
 mMiniDrone.setPitch((byte) 0);  
 mMiniDrone.setFlag((byte) 0);

// mMiniDrone.se  
 break;  
  
 default:  
  
 break;  
 }  
  
 }catch(Exception e){}  
 return true;  
 }  
 });

1. Go forward 1sec and wait for 5 sec and return back to your first position in 1 sec

findViewById(R.id.*search\_bt*).setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 try {  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
  
 mMiniDrone.setPitch((byte) 50);  
 mMiniDrone.setFlag((byte) 1);  
 Thread.*sleep*(1000);  
 mMiniDrone.setPitch((byte) 0);  
 mMiniDrone.setFlag((byte) 0);

Thread.*sleep*(5000);  
  
 mMiniDrone.setPitch((byte)-50);  
 mMiniDrone.setFlag((byte) 1);  
 Thread.*sleep*(1000);  
 mMiniDrone.setPitch((byte) 0);  
 mMiniDrone.setFlag((byte) 0);

// mMiniDrone.se  
 break;  
  
 default:  
  
 break;  
 }  
  
 }catch(Exception e){}  
 return true;  
 }  
 });

1. Forward 1sec , up .5 sec. down .5 sec
2. findViewById(R.id.*search\_bt*).setOnTouchListener(new View.OnTouchListener() {  
    @Override  
    public boolean onTouch(View v, MotionEvent event) {  
    try {  
    switch (event.getAction()) {  
    case MotionEvent.*ACTION\_DOWN*:  
     
    mMiniDrone.setPitch((byte) 50);  
    mMiniDrone.setFlag((byte) 1);  
    Thread.*sleep*(1000);  
    mMiniDrone.setPitch((byte) 0);  
    mMiniDrone.setFlag((byte) 0);  
     
    mMiniDrone.setGaz((byte) 50);  
    Thread.*sleep*(500);  
    mMiniDrone.setGaz((byte) 0);  
     
      
    mMiniDrone.setGaz((byte) -50);  
    Thread.*sleep*(500);  
    mMiniDrone.setGaz((byte) 0);  
     
   // mMiniDrone.se  
    break;  
     
    default:  
     
    break;  
    }  
     
    }catch(Exception e){}  
    return true;  
    }  
    });
3. Search : forward 1 sec, up .5 sec, rotate 5 sec, down .5 sec

findViewById(R.id.*search\_bt*).setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 try {  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
  
 mMiniDrone.setPitch((byte) 50);  
 mMiniDrone.setFlag((byte) 1);  
 Thread.*sleep*(1000);  
 mMiniDrone.setPitch((byte) 0);  
 mMiniDrone.setFlag((byte) 0);  
  
 mMiniDrone.setGaz((byte) 50);  
 Thread.*sleep*(500);  
 mMiniDrone.setGaz((byte) 0);  
  
  
 mMiniDrone.setYaw((byte) 50);  
 mMiniDrone.setFlag((byte) 1);  
 Thread.*sleep*(5000);  
 mMiniDrone.setYaw((byte) 0);  
 mMiniDrone.setFlag((byte) 0);  
  
 mMiniDrone.setGaz((byte) -50);  
 Thread.*sleep*(500);  
 mMiniDrone.setGaz((byte) 0);  
  
// mMiniDrone.se  
 break;  
  
 default:  
  
 break;  
 }  
  
 }catch(Exception e){}  
 return true;  
 }  
 });

<https://github.com/Parrot-Developers/Samples>