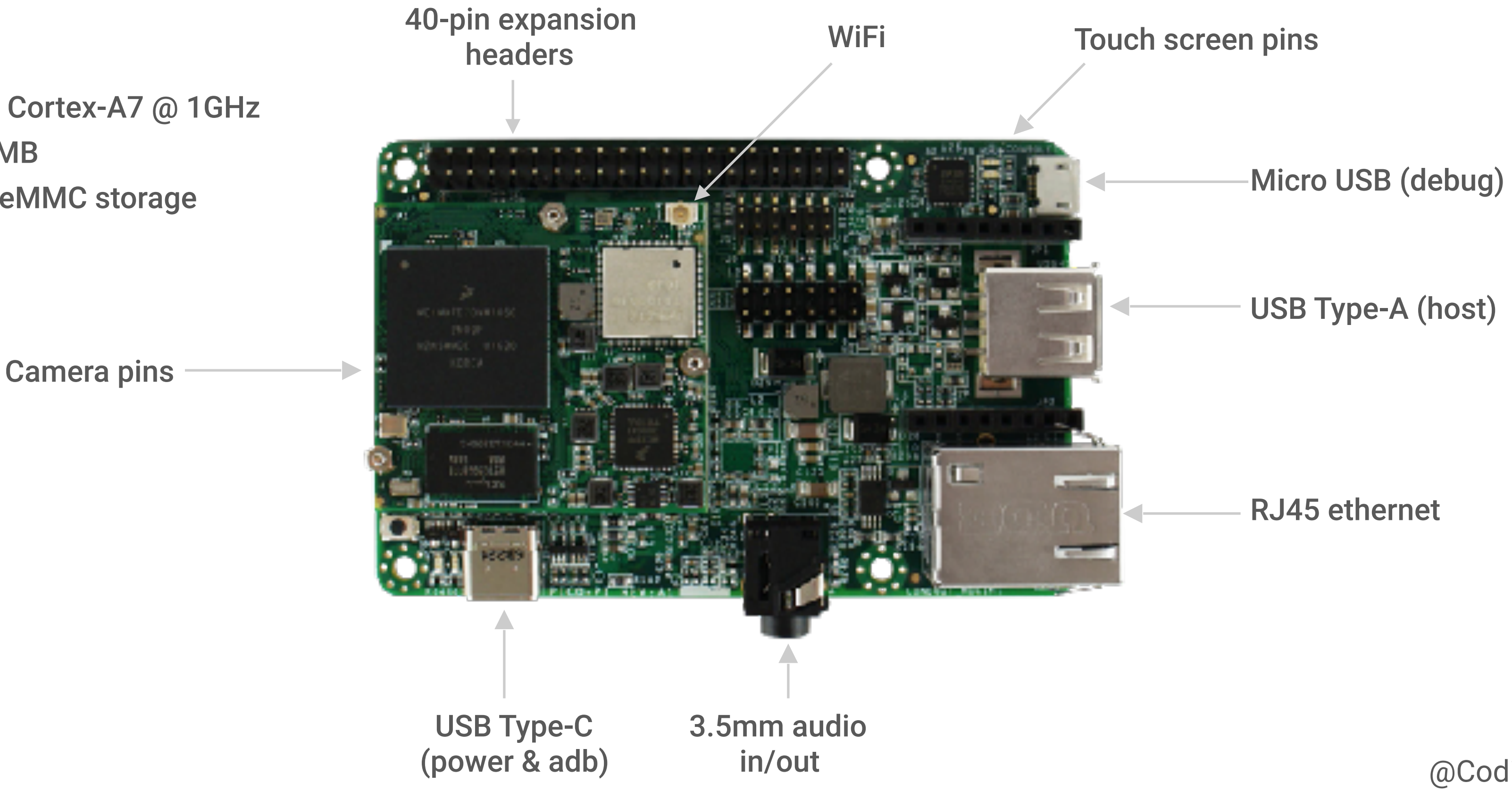
Connect your Android Things with Firebase

Doug Stevenson
@CodingDoug

android
things

PICO-PI-IMX7 — System on Module (SoM)

- ARM Cortex-A7 @ 1GHz
- 512 MB
- 4GB eMMC storage



@CodingDoug

Rainbow Hat peripheral

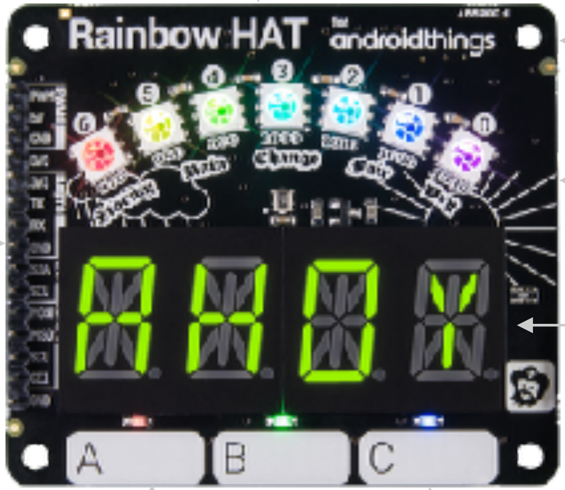
7 multicolor LEDs

Temp & pressure sensor

Piezo buzzer

Breakout pins for more peripherals

14-segment display



3 capacitative buttons

3 LEDs RGB

@CodingDoug

Types of peripheral I/O

General Purpose Input/Output (GPIO)

- Digital inputs and outputs with an on/off state.
- Buttons, relays, and proximity sensors.

Pulse Width Modulation (PWM)

- Variable control of a peripheral level.
- Servo motors, speakers, LEDs

Types of peripheral I/O

Inter-Integrated Circuit (I2C)

- Synchronous master serial bus allowing multiple slave devices addressed in software.
- Sensors, displays, advanced peripherals

Inter-IC Sound (I2S)

- Synchronous serial bus connecting digital sound peripherals that support PCM audio data.
- Digital microphones and digital-analog converters (DAC)

Types of peripheral I/O

Serial Peripheral Interface (SPI)

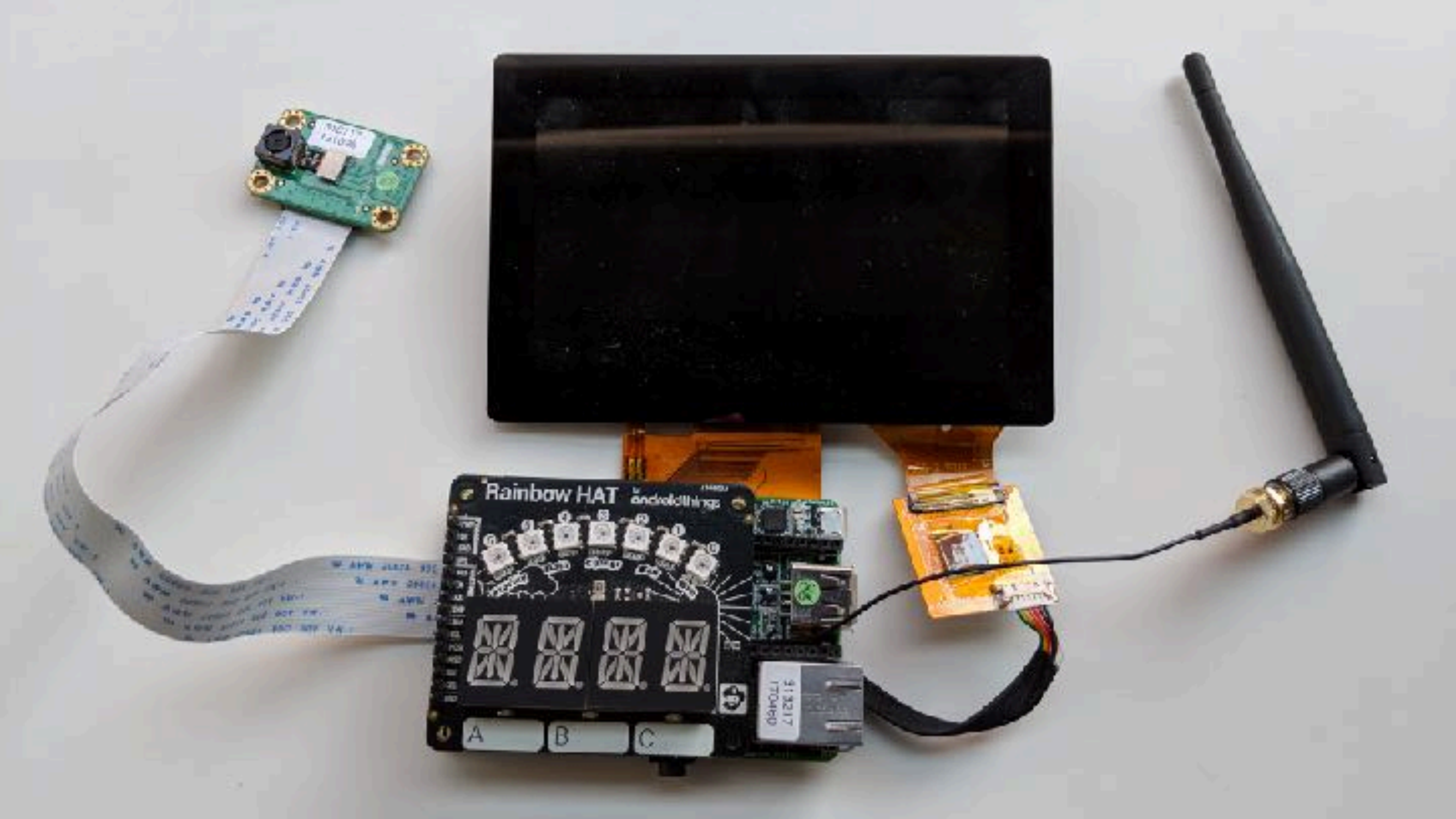
- Synchronous master serial bus allowing multiple slave devices addressed in hardware.
- Sensors, displays, higher speed peripherals

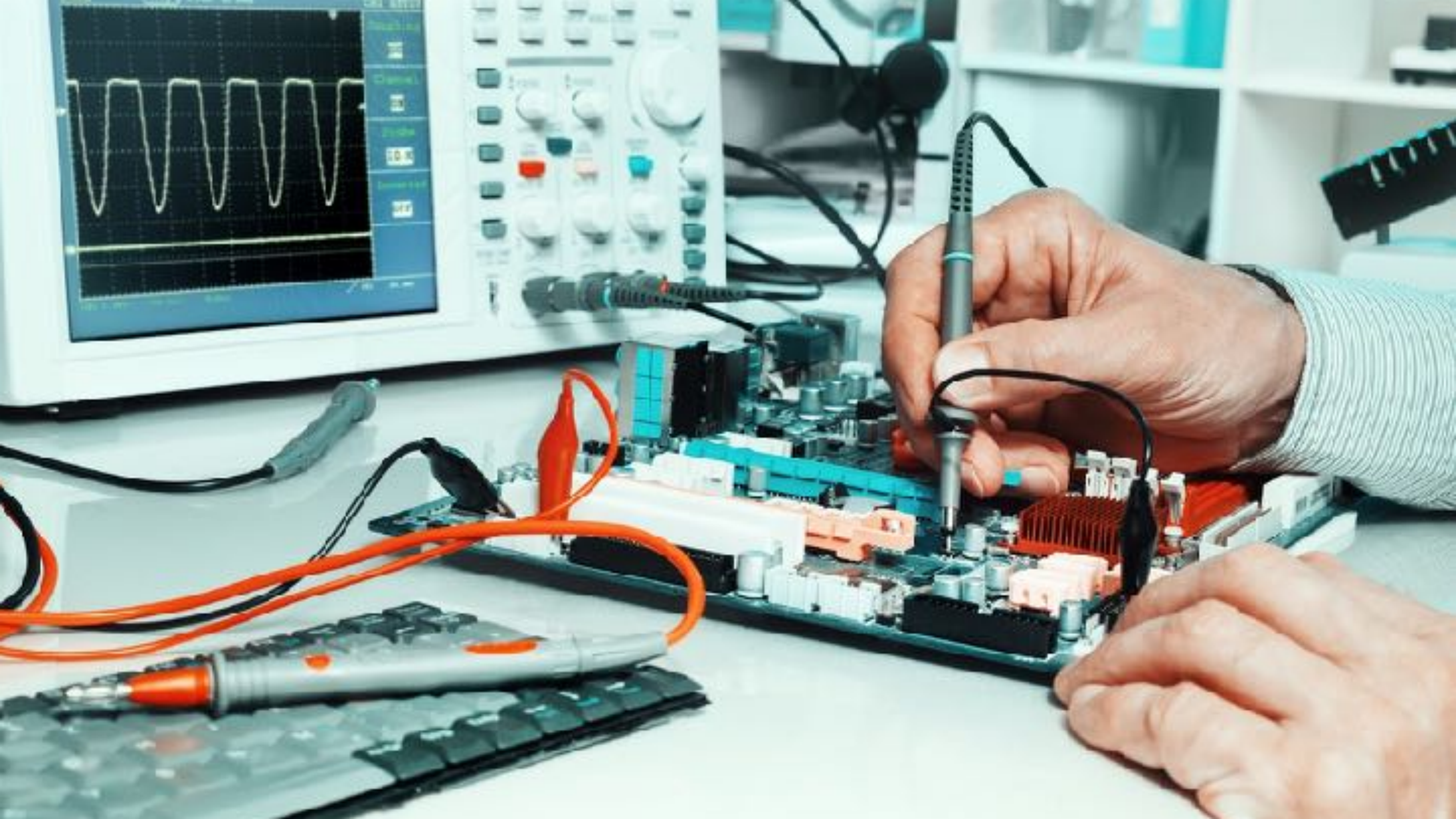
Universal Asynchronous Receiver Transmitter (UART)

- Asynchronous serial port used commonly in interrupt-driven applications.
- GPS, printers, RFID readers, barcode scanners

Android Things Drivers

```
dependencies {
    compileOnly 'com.google.android.things:androidthings:0.7-devpreview'

    implementation 'com.google.android.things.contrib:driver-rainbowhat:0.10'
    implementation ‘com.google.android.things.contrib:driver-button:0.6'
    implementation 'com.google.android.things.contrib:driver-bmx280:0.5'
    implementation 'com.google.android.things.contrib:driver-ht16k33:0.5'
    implementation 'com.google.android.things.contrib:driver-apa102:0.6'
    implementation ‘com.google.android.things.contrib:driver-pwmspeaker:0.4'
}
```


Rainbow HAT
androidthings
A
B
C



android
things

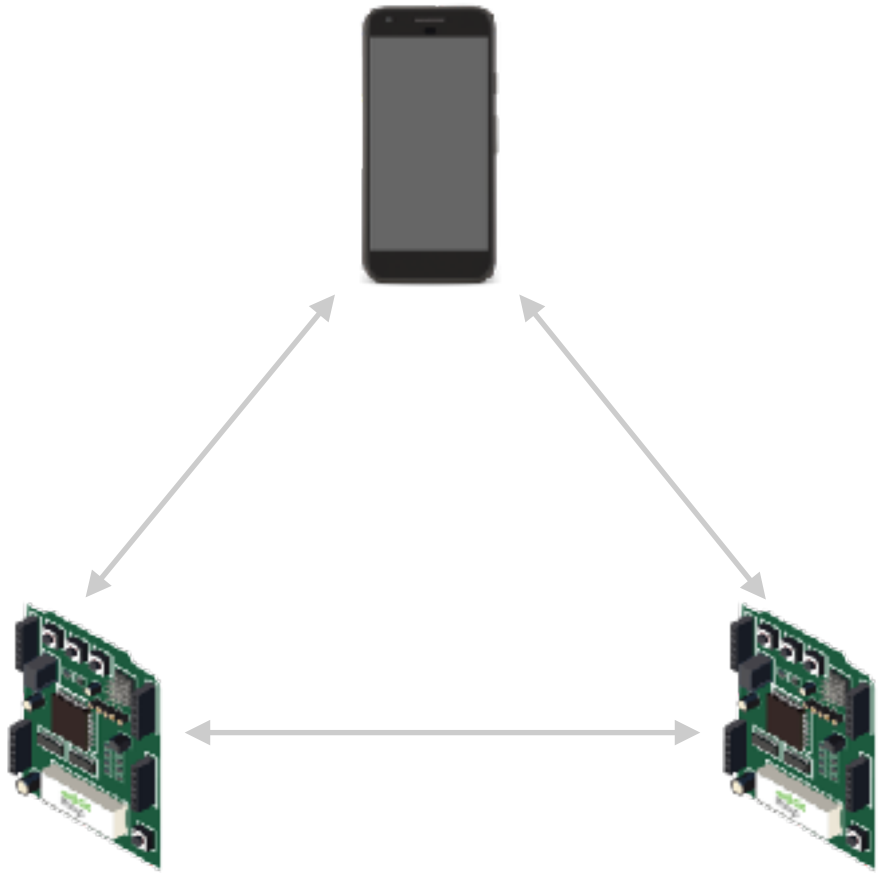
android
things

@CodingDoug

android things

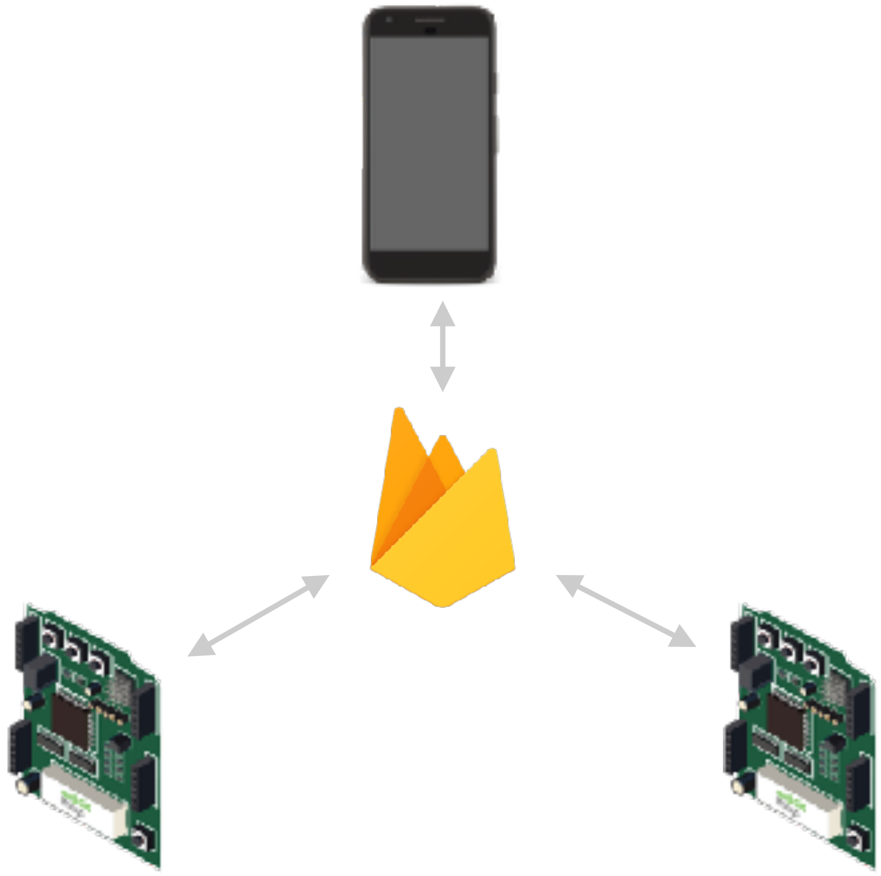
@CodingDoug

android things



@CodingDoug

android things



@CodingDoug

Firebase

Firebase

- Realtime Database
- Authentication
- Cloud Storage
- Test Lab
- Crashlytics

- Cloud Firestore
- Cloud Functions
- Hosting
- Performance
- Crash Reporting

- Analytics
- Dynamic Links
- Invites
- AdMob
- A/B Testing

- Cloud Messaging
- Remote Config
- App Indexing
- AdWords
- Predictions

Cloud Functions

for Firebase

JavaScript/TypeScript - node.js - express.js

@CodingDoug

android things

Cloud Firestore

Firebase Auth

Cloud Storage for Firebase

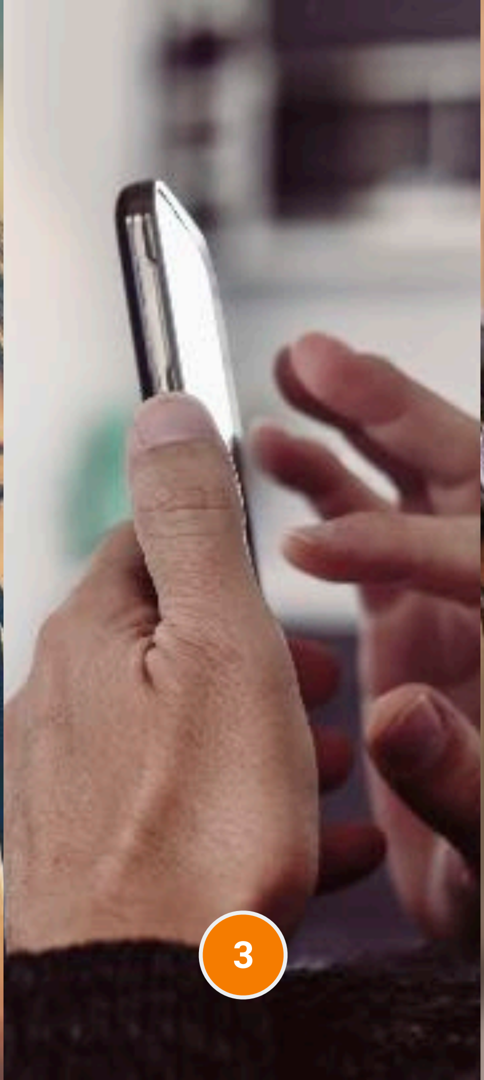
Firebase Cloud Messaging

Firebase Crashlytics

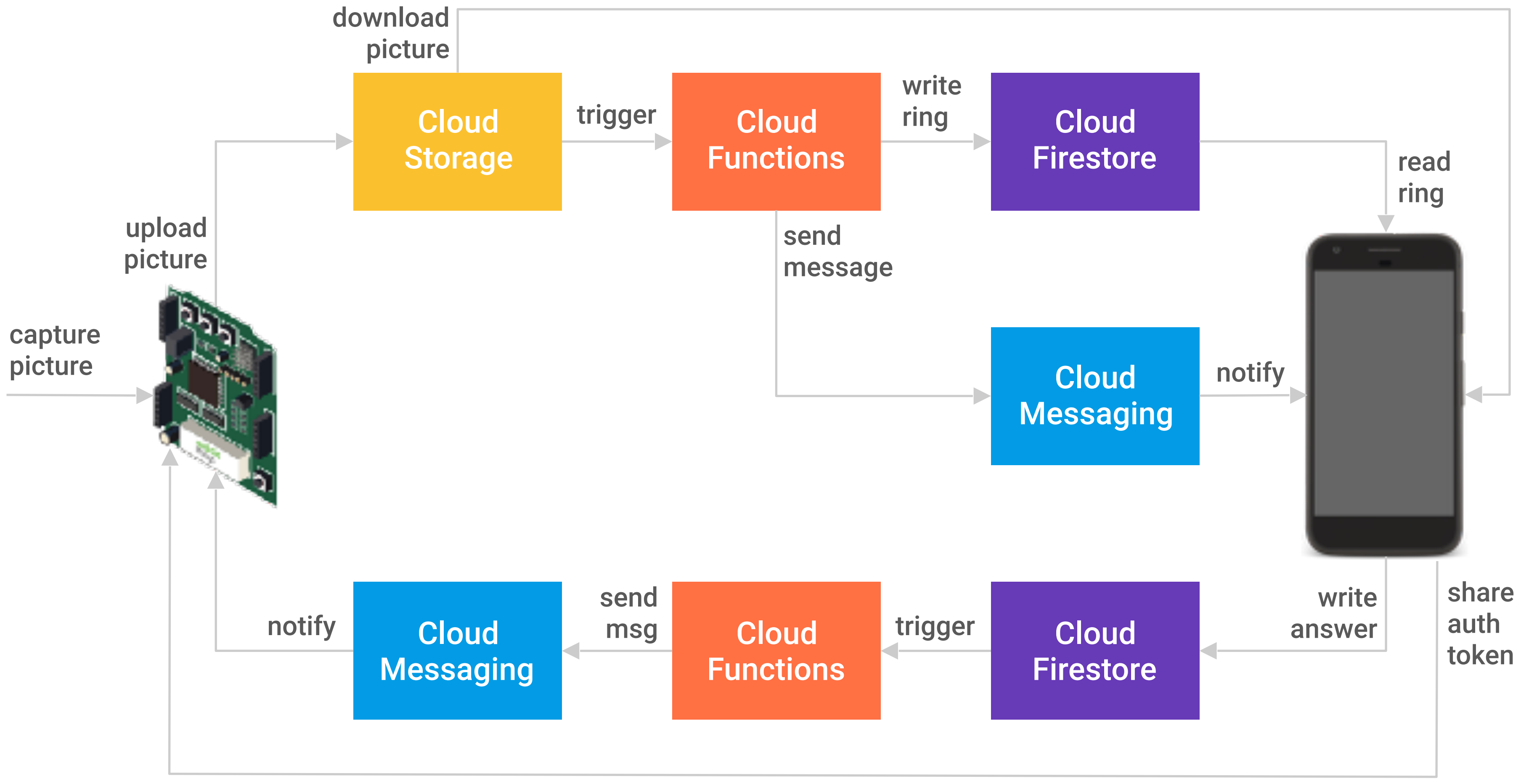
Cloud Functions for Firebase

1

2


3

4


download picture
Cloud Storage
trigger
Cloud Functions
write ring
Cloud Firestore
read ring
upload picture
send message
capture picture
Cloud Messaging
notify
write answer
share auth token
notify
Cloud Messaging
send msg
Cloud Functions
trigger
Cloud Firestore

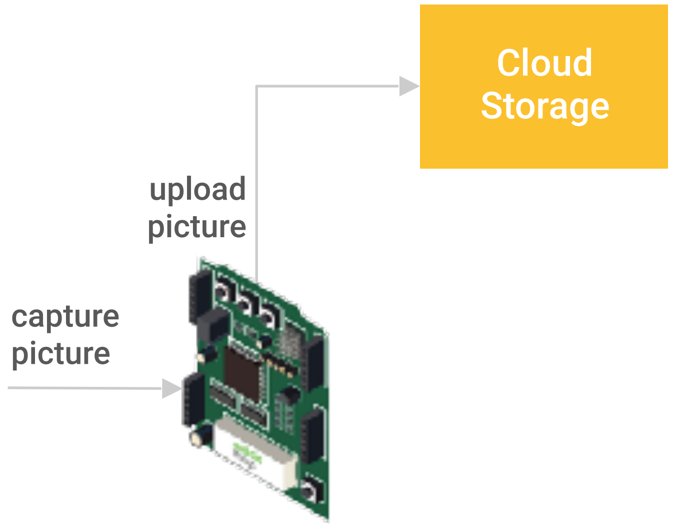
@CodingDoug

Capturing the picture

You can use the Android Camera2 API! goo.gl/mSDPm3

Issues I discovered:

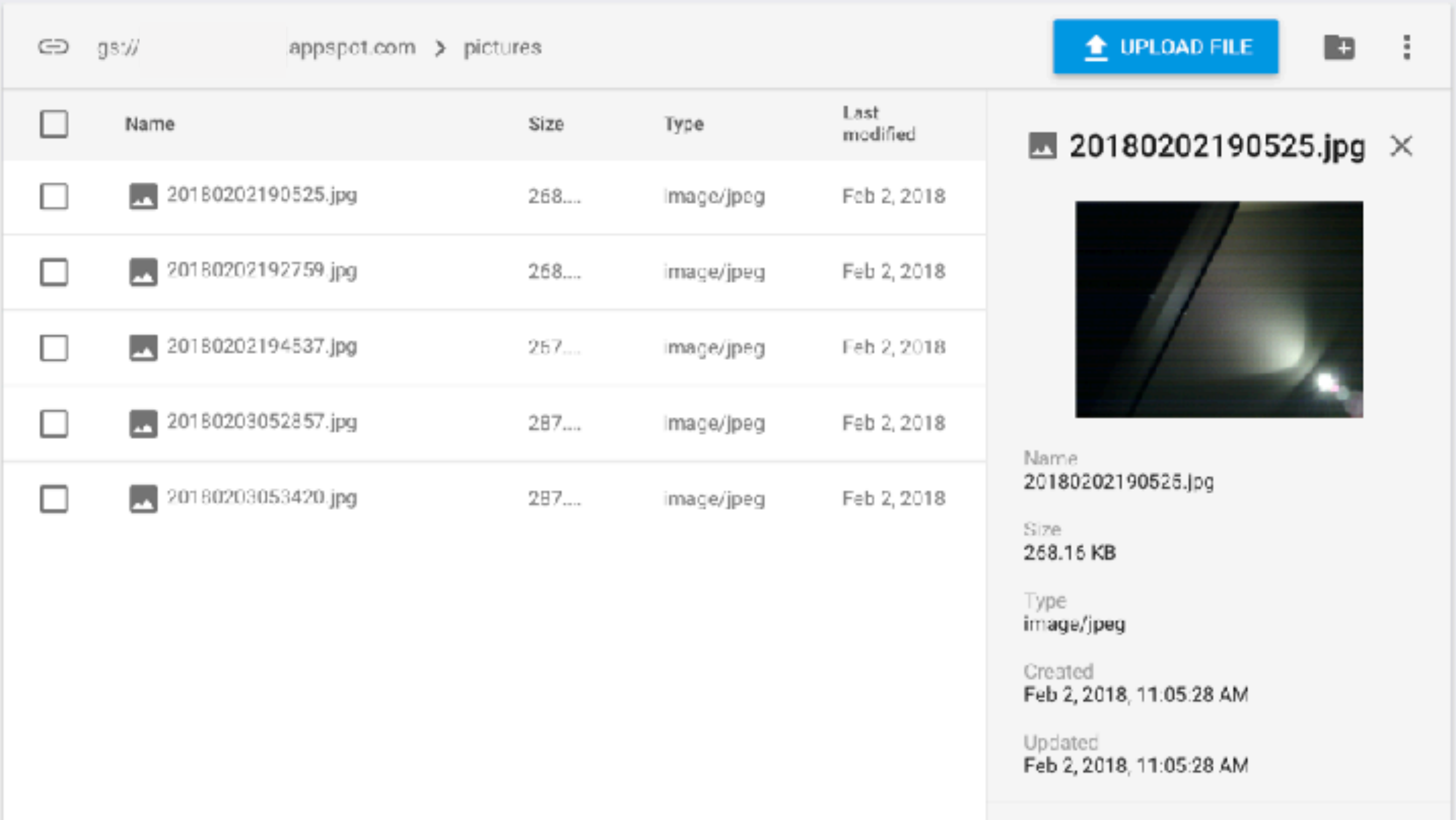
- In Developer Preview 0.5: TextureView not supported
- Ported to use SurfaceView instead
- Removed code that depends on autofocus
- In Developer Preview 0.6: TextureView now supported! https://goo.gl/XqTdXM
    - But it's slow without a GPU



@CodingDoug

Uploading to Cloud Storage

```
private fun uploadFile(file: File) {
    val sdf = SimpleDateFormat("yyyyMMddHHmmss", Locale.US)
    val storagePath = "/pictures/${sdf.format(Date())}.jpg"
    val ref = FirebaseStorage.getInstance().getReference(storagePath)
    ref.putFile(Uri.fromFile(file))
        .addOnSuccessListener(this) {
            Log.i(TAG, "Picture uploaded”)
        }
        .addOnFailureListener(this) { e ->
            Log.i(TAG, "Upload failed", e)
        }
        .addOnCompleteListener(this) {
            file.delete()
        }
}
```



gs:// appspot.com > pictures

UPLOAD FILE

	Name	Size	Type	Last modified
☐	20180202190525.jpg	268....	image/jpeg	Feb 2, 2018
☐	20180202192759.jpg	268....	image/jpeg	Feb 2, 2018
☐	20180202194537.jpg	267....	image/jpeg	Feb 2, 2018
☐	20180203052857.jpg	287....	image/jpeg	Feb 2, 2018
☐	20180203053420.jpg	287....	image/jpeg	Feb 2, 2018

20180202190525.jpg

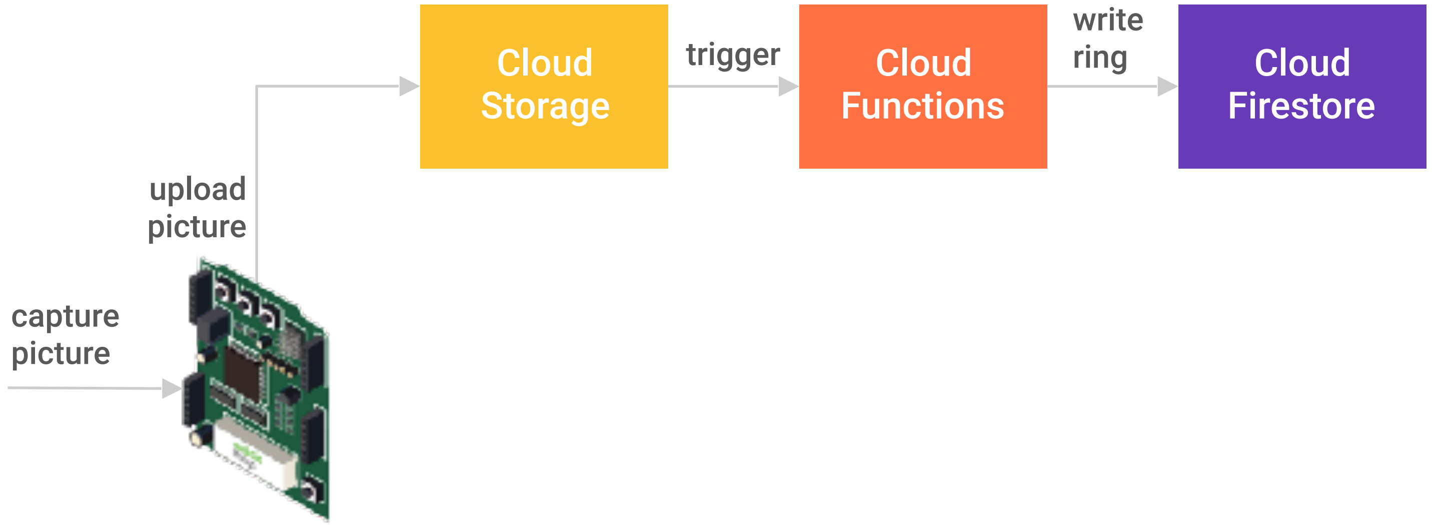
Name
20180202190525.jpg

Size
268.16 KB

Type
image/jpeg

Created
Feb 2, 2018, 11:05:28 AM

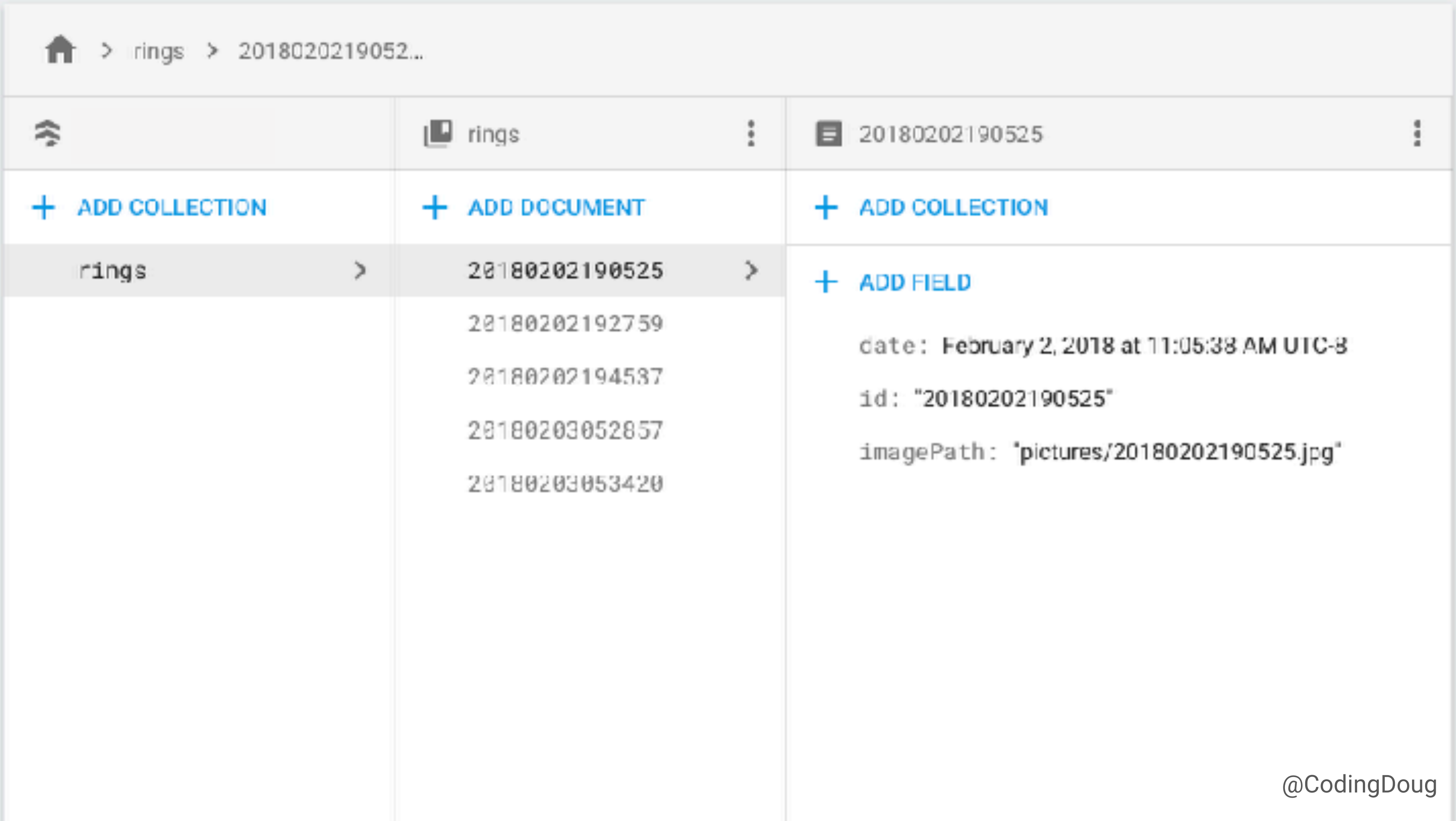
Updated
Feb 2, 2018, 11:05:28 AM


capture picture
upload picture
Cloud Storage
trigger
Cloud Functions
write ring
Cloud Firestore

@CodingDoug

Storage upload trigger pt. 1 - add document to Firestore

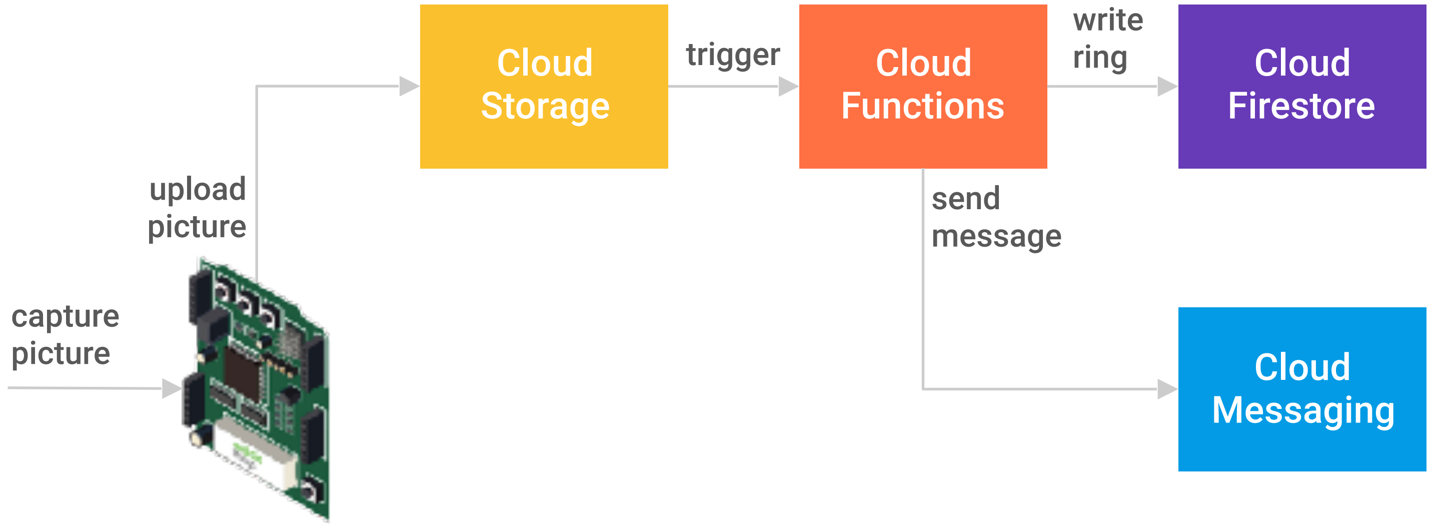
```
export const onRing = functions.storage.object().onChange(_onRing)
async function _onRing(event: functions.Event<ObjectMetadata>): Promise<any> {
    const path = event.data.name        // e.g. /pictures/20180327123000.jpg
    const id = basename(path, '.jpg')   // e.g. 20180327123000

    try {
        // Add a document to Firestore with the details of this ring
        //
        const ring: Ring = {
            id: id,
            date: new Date(),
            imagePath: path,
        }

        await firestore.collection('rings').doc(id).set(ring)
```



```
20180202190525
20180202192759
20180202194537
20180203052857
20180203053420
```

```
date: February 2, 2018 at 11:05:38 AM UTC-8
id: "20180202190525"
imagePath: "pictures/20180202190525.jpg"
```

@CodingDoug

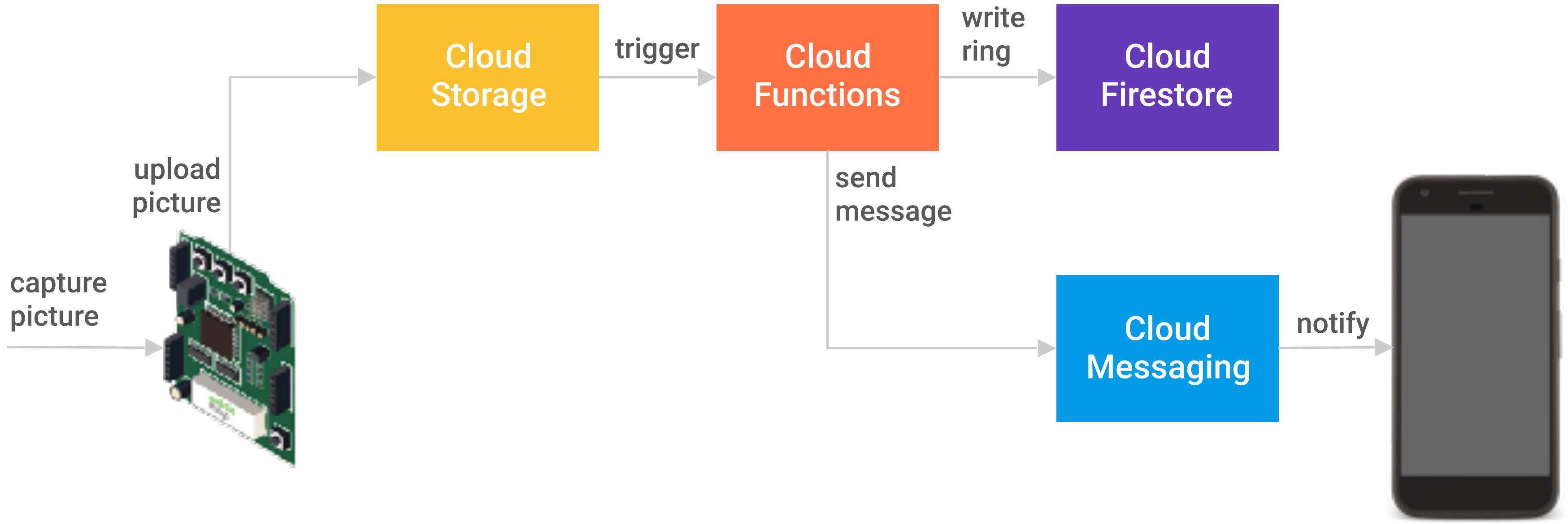


Cloud Storage
trigger
Cloud Functions
write ring
Cloud Firestore
upload picture
send message
capture picture
Cloud Messaging

@CodingDoug

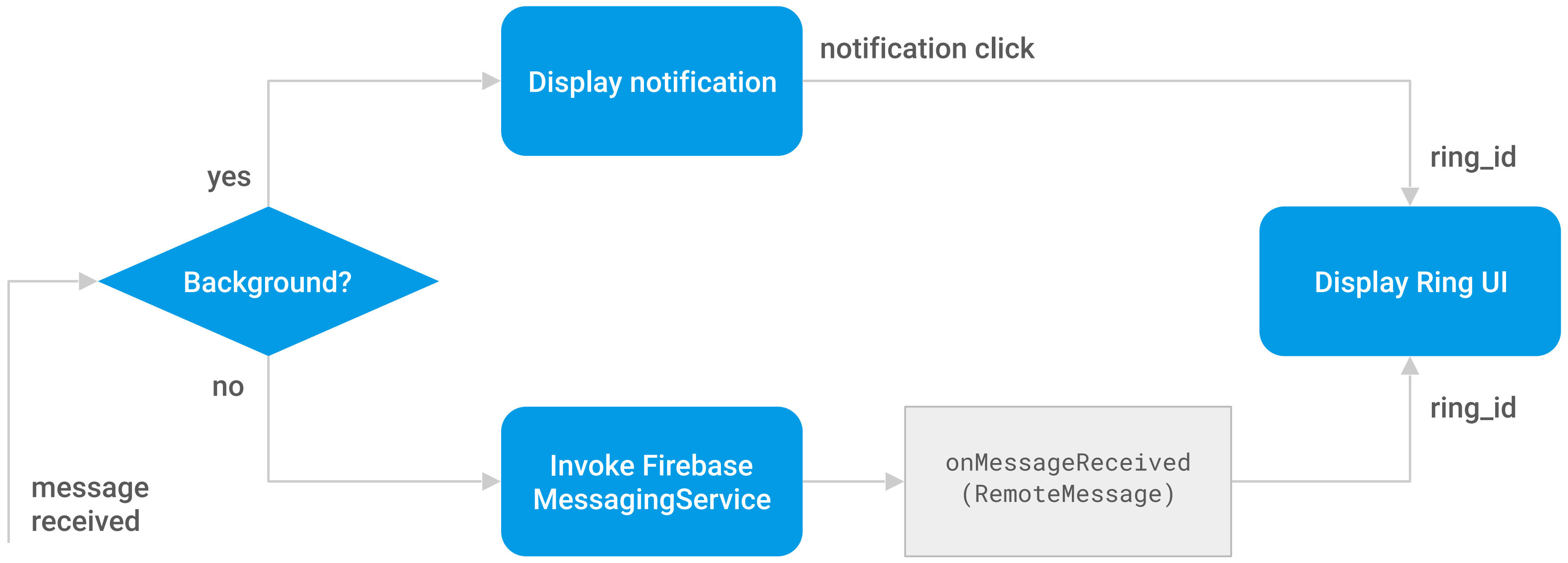
Storage upload trigger pt. 2 - send notification to app

```
// Send a notification to the app
//
const payload = {
    notification: {
        title: 'Ring Ring!',
        body: 'There is someone at the door!',
        click_action: 'com.hyperaware.doorbell.ANSWER_RING'
    },
    data: {
        ring_id: id
    }
}

const response = await fcm.sendToTopic('rings', payload)
```


capture picture
upload picture
Cloud Storage
trigger
Cloud Functions
write ring
Cloud Firestore
send message
Cloud Messaging
notify

@CodingDoug

Receiving the notification



Earlier: Subscribe to “rings” topic

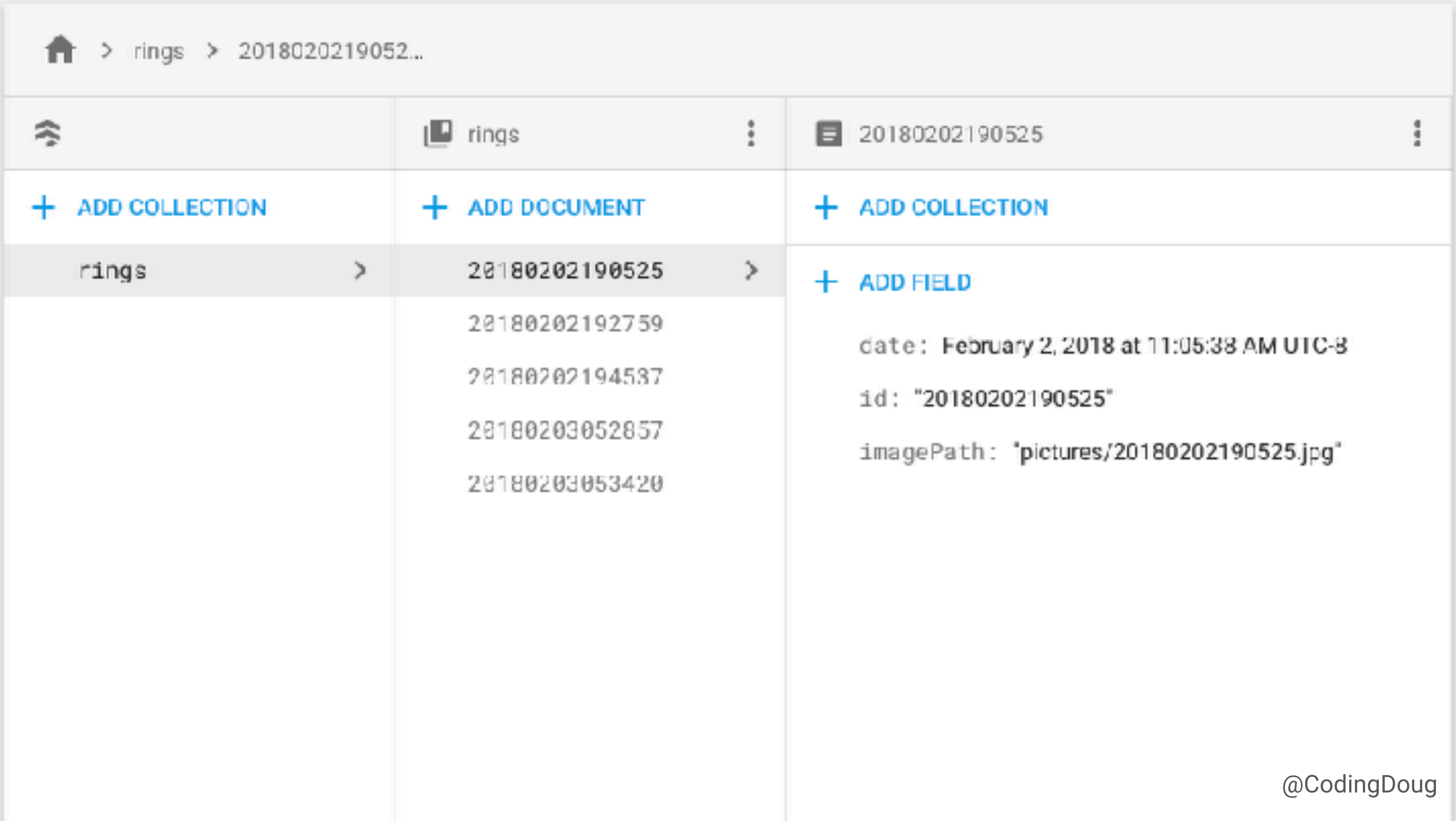
```
class MyInstanceIdService : FirebaseInstanceIdService() {

    companion object {
        private const val TAG = "MyInstanceIdService"
    }

    override fun onTokenRefresh() {
        super.onTokenRefresh()
        Log.d(TAG, "FCM token refresh: ${FirebaseInstanceId.getInstance().token!!}")
        FirebaseMessaging.getInstance().subscribeToTopic("rings")
    }

}
```

Handle incoming ring data message

```
class OnRingMessagingService : FirebaseMessagingService() {

    override fun onMessageReceived(remoteMessage: RemoteMessage) {
        super.onMessageReceived(remoteMessage)
        if (remoteMessage.data.containsKey("ring_id")) {
            val ringId = remoteMessage.data["ring_id"]
            val intent = Intent(this, AnswerRingActivity::class.java)
            intent.putExtra("ring_id", ringId)
            startActivity(intent)
        }
        else {
            Log.w(TAG, "Data message received without ring_id")
        }
    }
}
```

Handle incoming ring (Activity)

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    val extras = intent.extras
    if (extras == null) {
        Log.e(TAG, "ring_id was not provided")
        finish()
        return
    }

    val ringId = extras.getString("ring_id")
    if (ringId.isEmpty()) {
        Log.e(TAG, "ring_id was empty")
        finish()
        return
    }

    // display it...
}
```

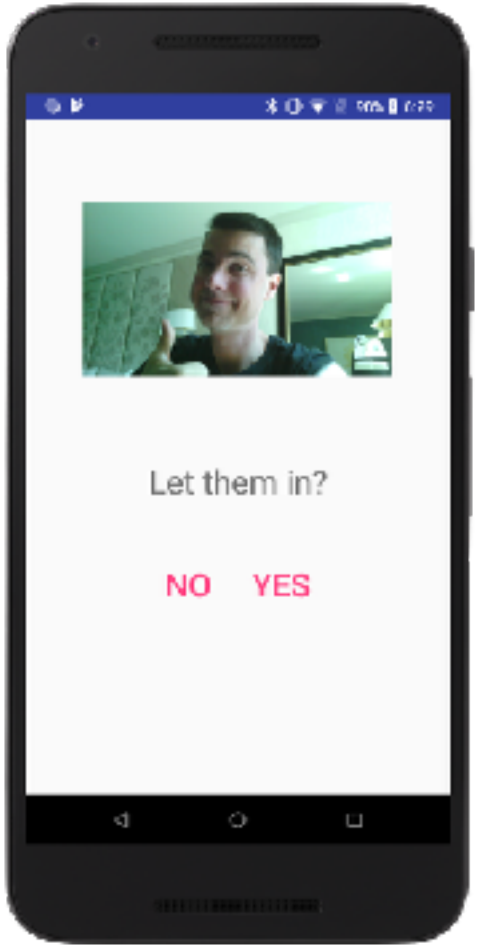


rings

20180202190525
20180202192759
20180202194537
20180203052857
20180203053420

```
date: February 2, 2018 at 11:05:38 AM UTC-8
id: "20180202190525"
imagePath: "pictures/20180202190525.jpg"
```

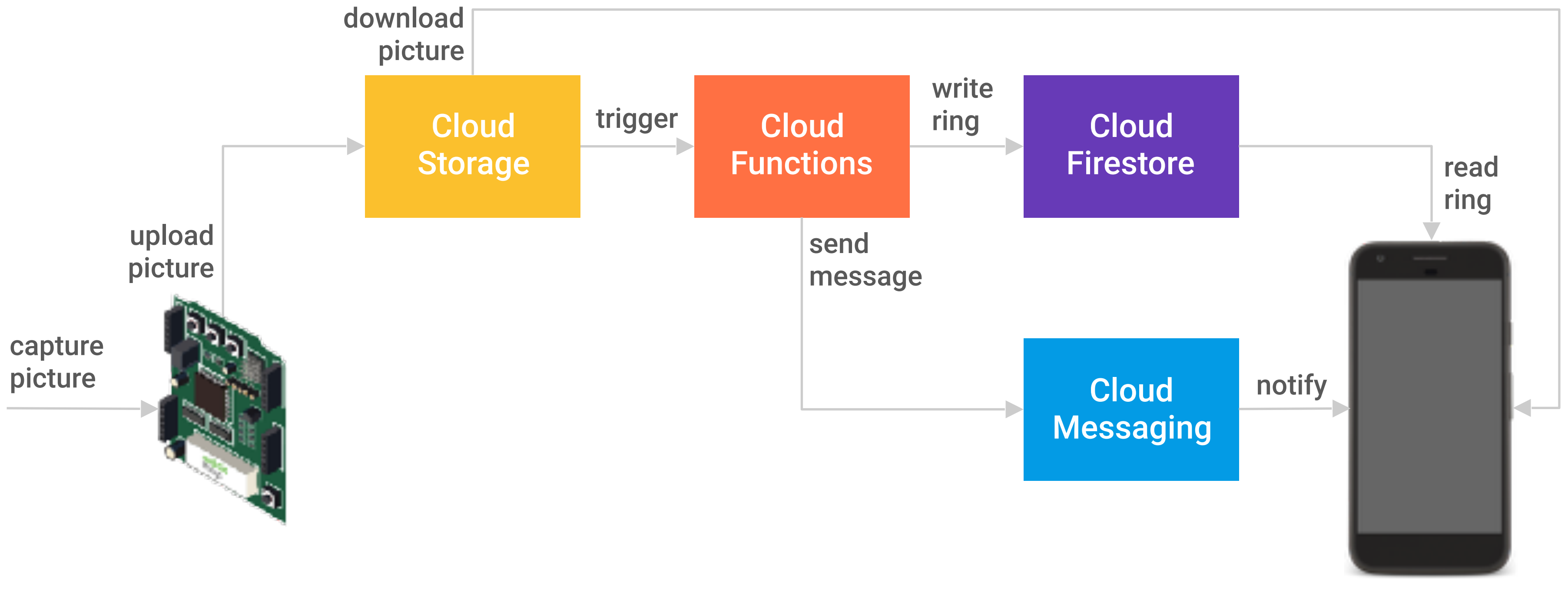
@CodingDoug


Let them in?

NO YES

Firebase

@CodingDoug


download picture
Cloud Storage
trigger
Cloud Functions
write ring
Cloud Firestore
read ring
upload picture
send message
capture picture
Cloud Messaging
notify

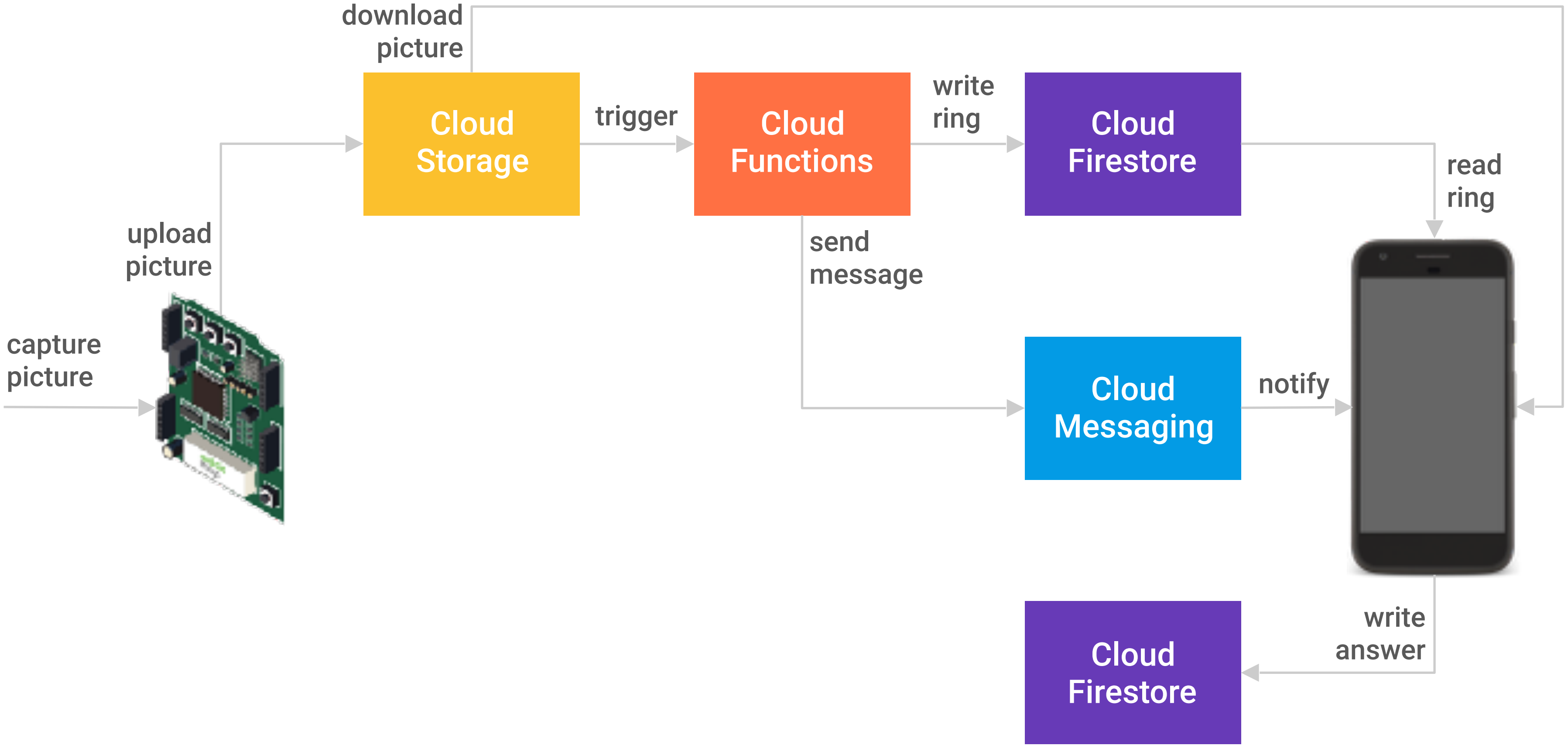
@CodingDoug

Fetch ring data from Firestore

```
private fun populateViews(ringId: String) {
    ringReference = FirebaseFirestore.getInstance().collection("rings").document(ringId)
    ringReference.get()
        .addOnSuccessListener(this) { snap ->
            if (snap.exists()) {
                val ring = snap.toObject(Ring::class.java)
                val ref = FirebaseStorage.getInstance().getReference(ring.imagePath!!)
                Glide.with(this@AnswerRingActivity).load(ref).into(ivGuest)
            }
        }
        .addOnFailureListener(this) { error ->
            Log.e(TAG, "Can't fetch ring $ringId", error)
        }
}
```

Glide Module — Cloud Storage for Firebase plugin

```
@GlideModule
class MyAppGlideModule : AppGlideModule() {

    override fun registerComponents(context: Context, glide: Glide, registry: Registry) {
        // Register FirebaseImageLoader to handle StorageReference
        registry.append(StorageReference::class.java, InputStream::class.java,
            FirebaseImageLoader.Factory())
    }

}
```

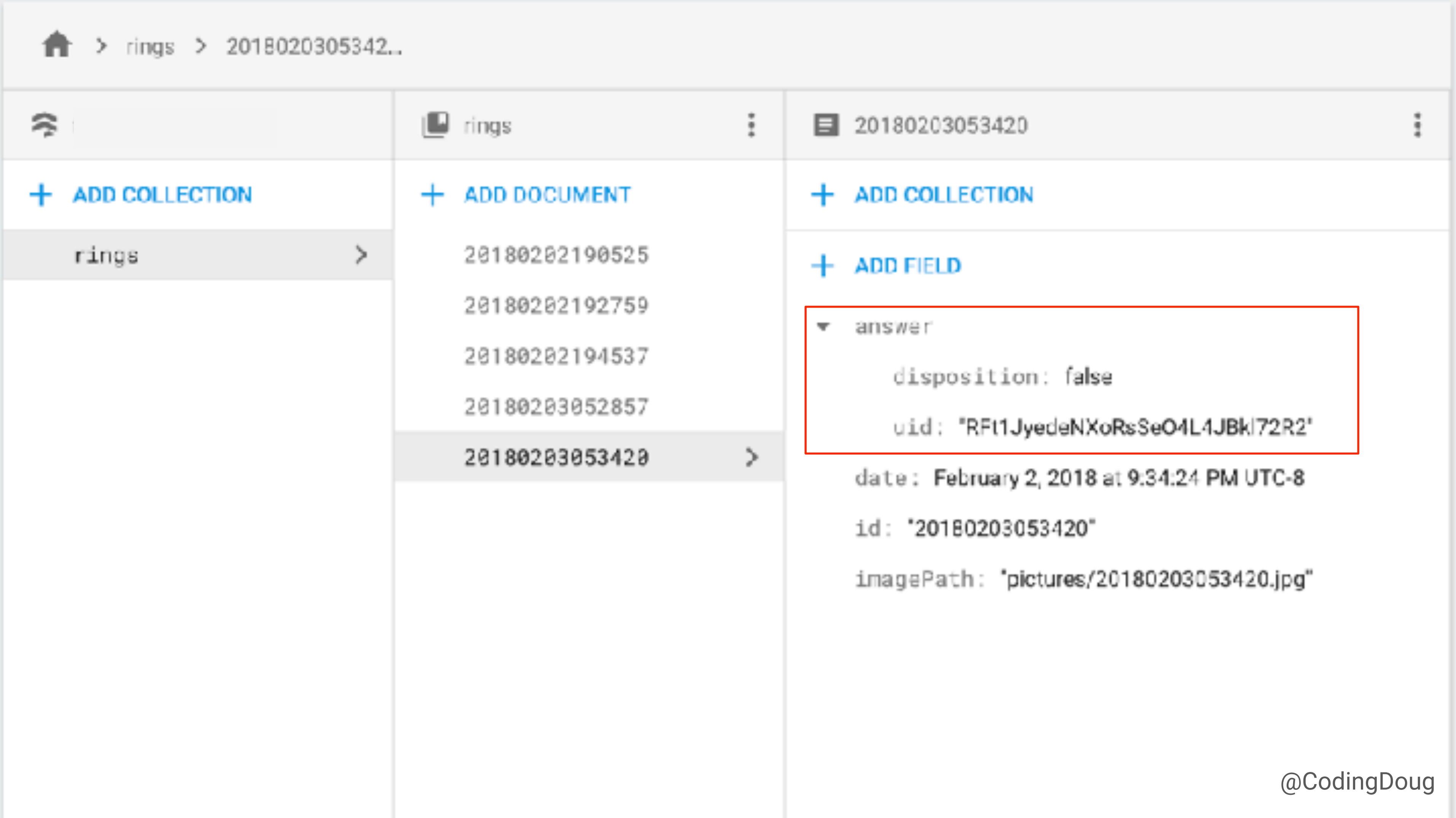


download picture
Cloud Storage
trigger
Cloud Functions
write ring
Cloud Firestore
read ring
upload picture
send message
capture picture
Cloud Messaging
notify
write answer
Cloud Firestore

@CodingDoug

Update ring disposition in Firestore

```
val disposition = button_click_true_or_false
ringReference.update(
    "answer.uid", uid,
    "answer.disposition", disposition)
    .addOnCompleteListener(this) {
        Log.d(TAG, "Answer written to database")
        finish()
    }
    .addOnFailureListener(this, { e ->
        Log.d(TAG, "Answer not written to database", e)
        finish()
    })
```



rings > 2018020305342...

rings

20180203053420

ADD COLLECTION

ADD DOCUMENT

ADD COLLECTION

rings

20180202190525

20180202192759

20180202194537

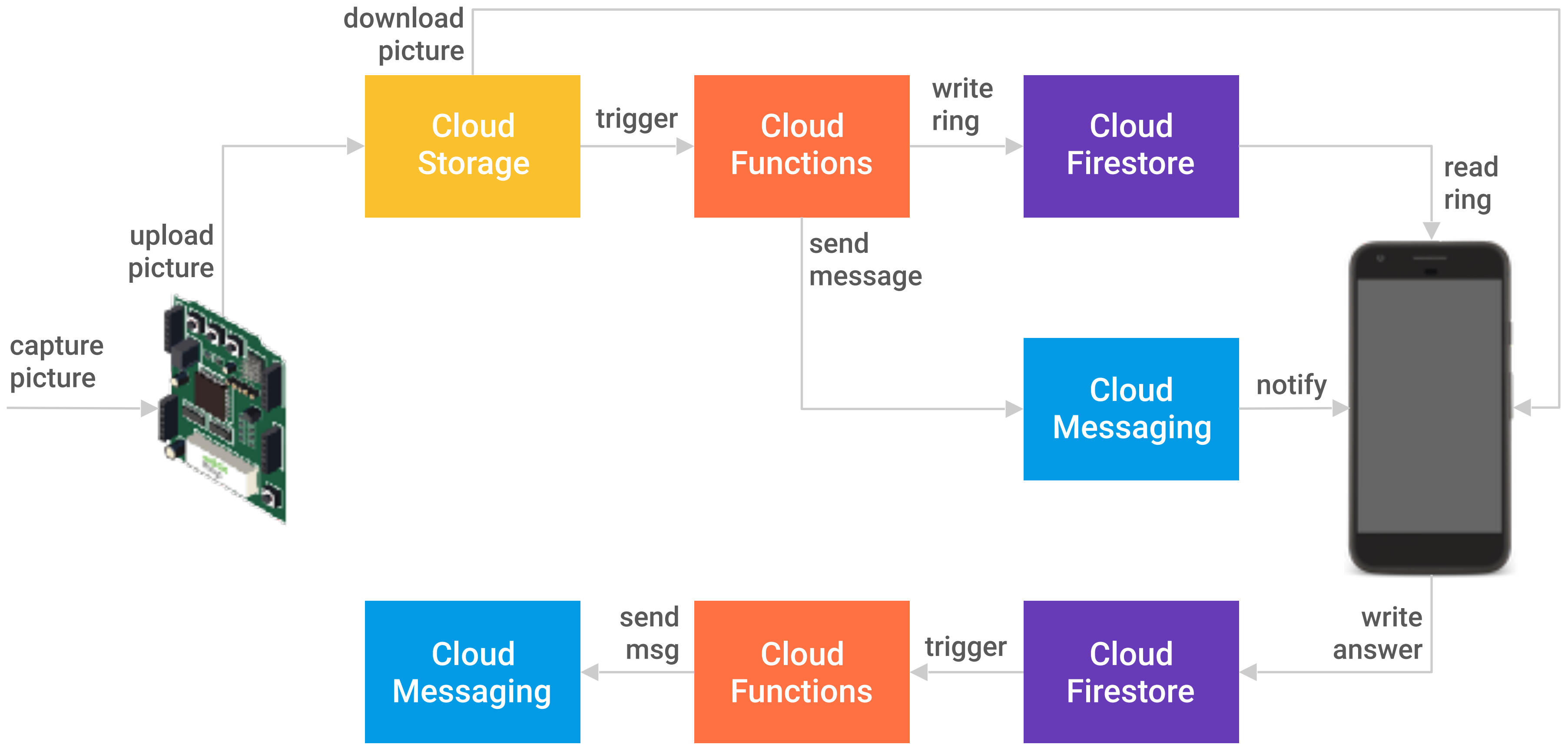
20180203052857

20180203053420

ADD FIELD

```
answer
    disposition: false
    uid: "RFt1JyedeNXoRsSeO4L4JBkl72R2"
date: February 2, 2018 at 9:34:24 PM UTC-8
id: "20180203053420"
imagePath: "pictures/20180203053420.jpg"
```

@CodingDoug


download picture
Cloud Storage
trigger
Cloud Functions
write ring
Cloud Firestore
read ring
upload picture
send message
capture picture
Cloud Messaging
notify
write answer
Cloud Messaging
send msg
Cloud Functions
trigger
Cloud Firestore

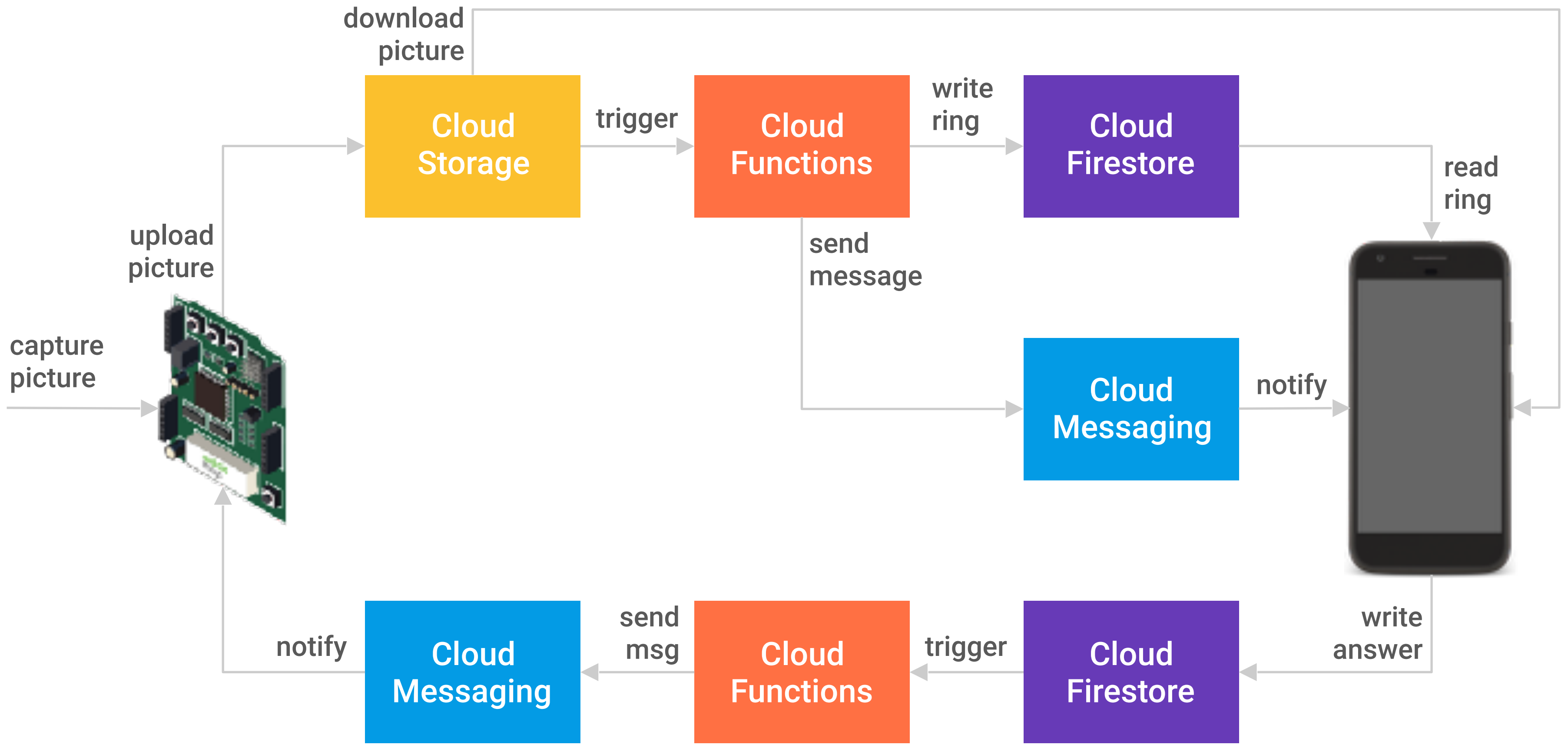
@CodingDoug

Firestore trigger pt. 1 — send answer to Android Thing

```
export const onAnswer = functions.firestore.document('/rings/{ringId}').onUpdate(_onAnswer)
async function _onAnswer(event: functions.Event<DeltaDocumentSnapshot>): Promise<any> {
    const ringId = event.params.ringId
    const previous = event.data.previous.data() as Ring
    const ring = event.data.data() as Ring

    // Only interested in rings that have a new answer
    if (previous.answer || !ring.answer) {
        console.log("This is not the update you're looking for.")
        return Promise.resolve()
    }

// cont'd...
```

Firestore trigger pt. 2 — send answer to Android Thing

```
    const payload = {
        data: {
            disposition: ring.answer.disposition.toString(),
            ring_id: ringId
        }
    }
    try {
        const response = await fcm.sendToTopic('answers', payload)
        console.log(`ring ${ringId} answer sent:`, response)
    }
    catch (err) {
        console.error(`ring ${ringId} answer error:`, err)
    }
}
```


download picture
Cloud Storage
trigger
Cloud Functions
write ring
Cloud Firestore
read ring
upload picture
send message
capture picture
Cloud Messaging
notify
write answer
notify
Cloud Messaging
send msg
Cloud Functions
trigger
Cloud Firestore

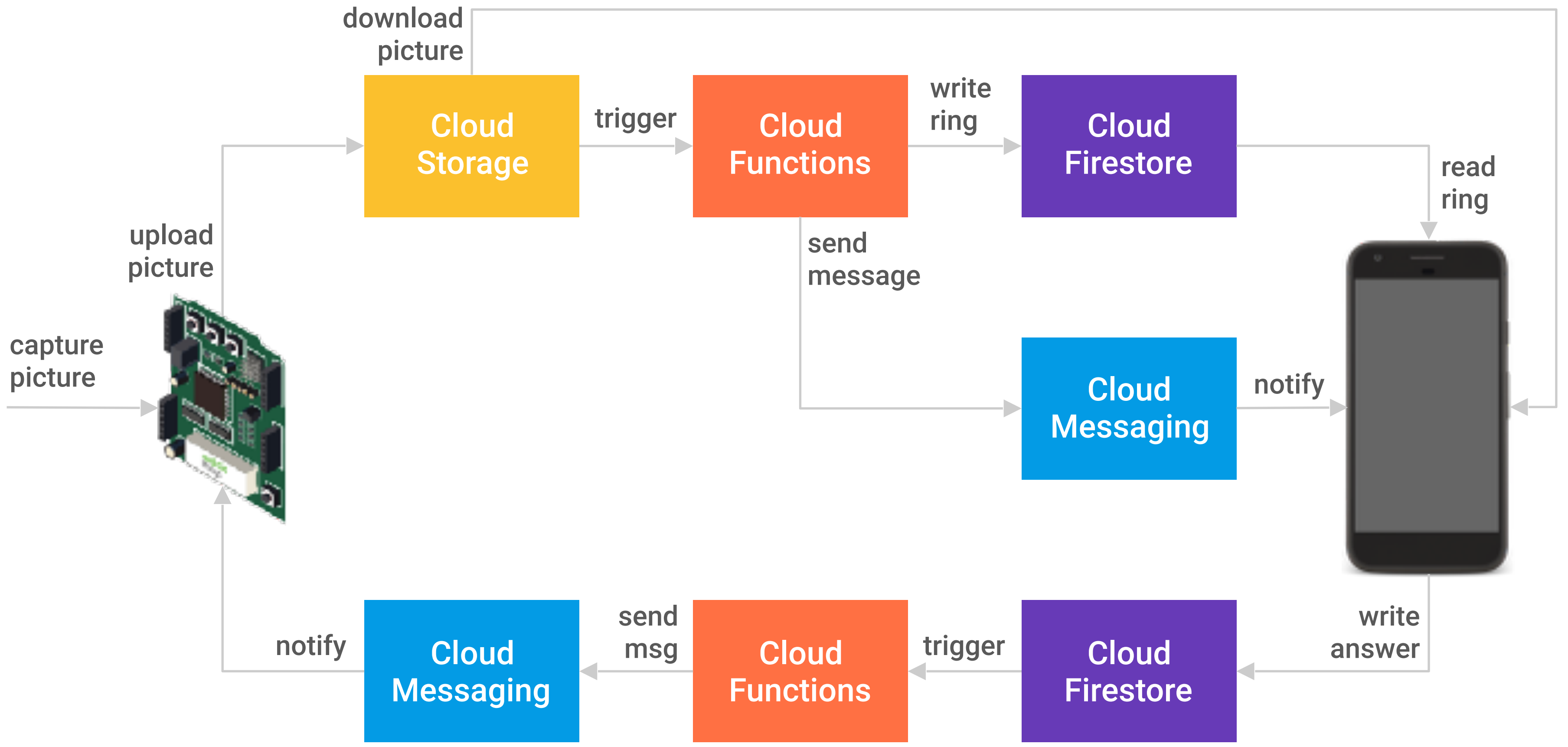
@CodingDoug

On Thing, earlier: Subscribe to “answers” topic

```
class MyInstanceIdService : FirebaseInstanceIdService() {

    companion object {
        private const val TAG = "MyInstanceIdService"
    }

    override fun onTokenRefresh() {
        super.onTokenRefresh()
        Log.d(TAG, "FCM token refresh: ${FirebaseInstanceId.getInstance().token!!}")
        FirebaseMessaging.getInstance().subscribeToTopic("answers")
    }

}
```

Handle incoming answer data message

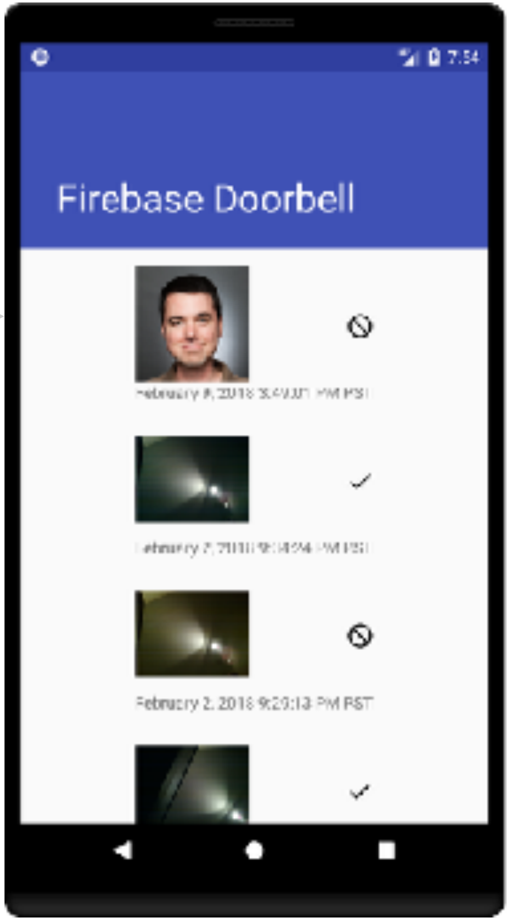
```
class OnAnswerMessagingService : FirebaseMessagingService() {

    override fun onMessageReceived(remoteMessage: RemoteMessage) {
        super.onMessageReceived(remoteMessage)
        if (remoteMessage.data.containsKey("disposition")) {
            val d = java.lang.Boolean.parseBoolean(remoteMessage.data["disposition"])
            val intent = Intent(this, ResponseActivity::class.java)
            intent.putExtra("disposition", d)
            startActivity(intent)
        }
        else {
            Log.w(TAG, "Data message received without disposition")
        }
    }
}
```


download picture
Cloud Storage
trigger
Cloud Functions
write ring
Cloud Firestore
read ring
upload picture
send message
capture picture
Cloud Messaging
notify
write answer
notify
Cloud Messaging
send msg
Cloud Functions
trigger
Cloud Firestore

@CodingDoug

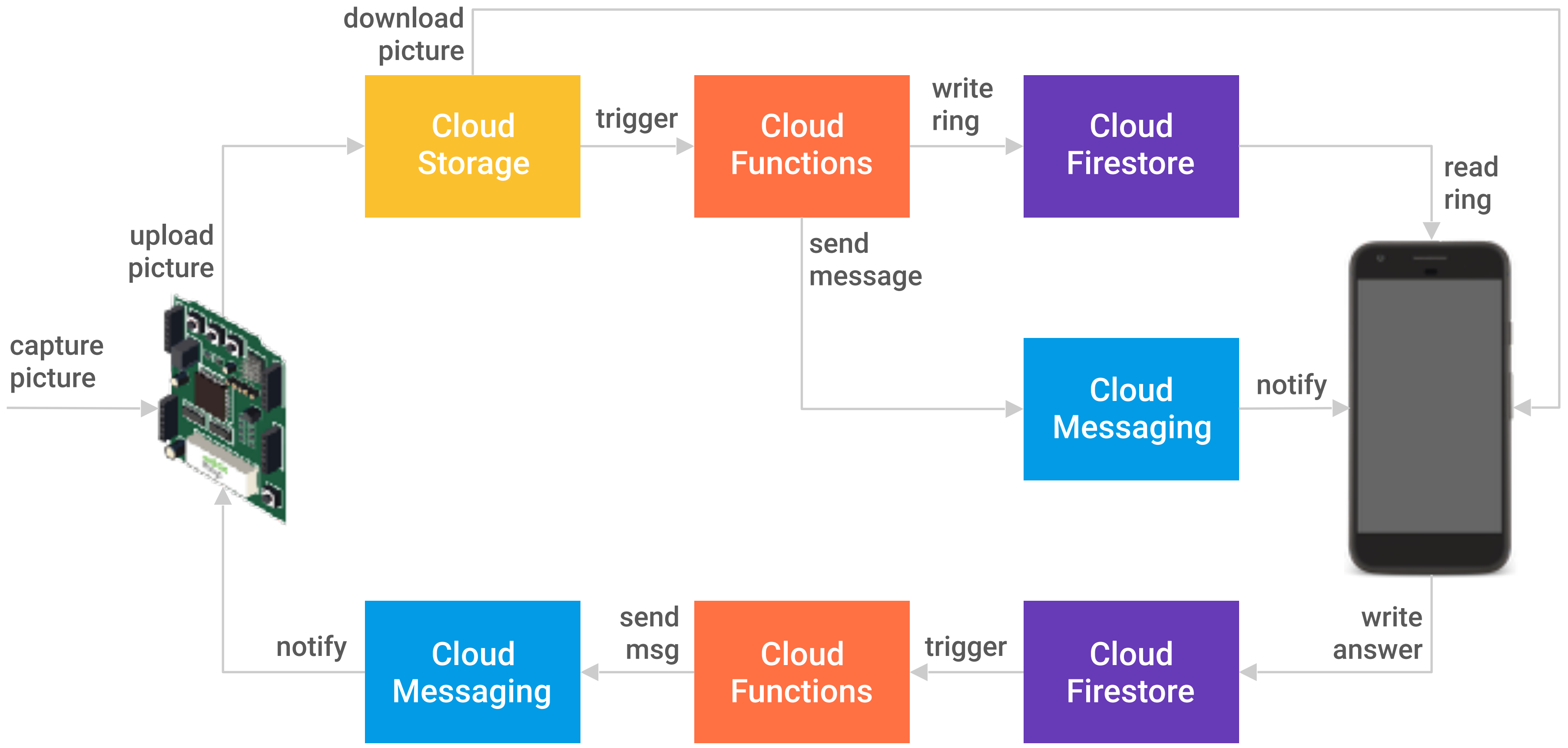
This is a RecyclerView

FirestoreRecyclerAdapter makes this super easy!
FirebaseUI FTW
https://goo.gl/oQsk4h



February 2, 2018 9:20:13 PM PST

What about security?


download picture
Cloud Storage
trigger
Cloud Functions
write ring
Cloud Firestore
read ring
upload picture
send message
capture picture
Cloud Messaging
notify
write answer
notify
Cloud Messaging
send msg
Cloud Functions
trigger
Cloud Firestore

@CodingDoug

Cloud St   ge security rules — universal read and write

```
ser       rebase.sto      {
    h     bucket}/o
    atch     Paths=**
      allow       write    f true;

  }
}
```

Firesto[illegible]curity rules — universal read and write

```
ser      loud.firest      {
    ch      abases/{da      se}/documents {
     atch      ument=**
      allow      write      f true;

  }
}
```

What to do about security?

- Yeah, it’s easy to get started without security
- All reads and writes should require user authentication (minimally)
- Also consider data validation rules
- See also for Storage: https://firebase.google.com/docs/storage/security/
- See also for Firestore: https://firebase.google.com/docs/firestore/security/get-started

Authentication login flows are hard (and boring)

Authentication is easy (with FirebaseUI)

```
implementation "com.firebaseui:firebase-ui-auth:$firebase_ui_version"
```

App login with FirebaseUI — launch UI flow

```
findViewById<Button>(R.id.btn_sign_in).setOnClickListener {
    startActivityForResult(
        AuthUI.getInstance()
            .createSignInIntentBuilder()
            .setAvailableProviders(listOf(AuthUI.IdpConfig.GoogleBuilder().build()))
            .build(),
        RC_SIGN_IN)
}
```

App login with FirebaseUI — handle login results

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == RC_SIGN_IN) {
        val response = IdpResponse.fromResultIntent(data)
        if (resultCode == Activity.RESULT_OK) {
            // handle login
        }
    }
}
```

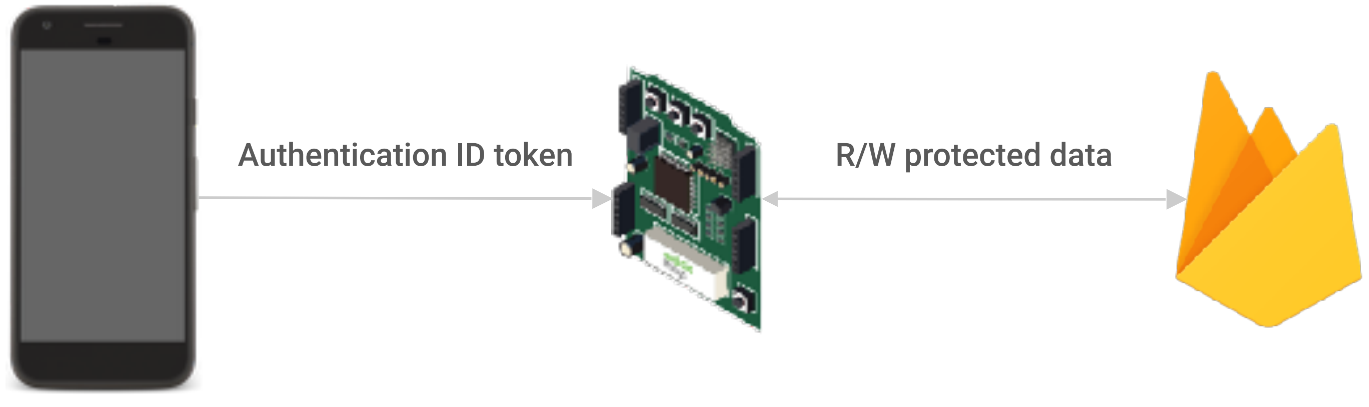
Firebase

@CodingDoug

How does a Thing get logged in?

- UI for various provider logins aren’t supported on Android Things
- Android Thing may not even have a screen!
- Anonymous login works, not a great solution
- Best to get dedicated user credentials for a real login

Share an auth credential from app to Thing



But how?

Nearby API

https://developers.google.com/nearby/

@CodingDoug

Nearby Messages

- Peer-to-peer pub/sub messaging model
- Device pairing via combo of Wifi, BT, BLE, near-ultrasonic radio
- Data payload exchange via Google server & Cloud project
- Didn’t work with Android Things 0.6.1
- Now works in 0.7.0!

Nearby Connections

- Peer-to-peer networking, high bandwidth, low latency
- Data exchange via Wifi, BT, BLE (no internet required)

Publish a Nearby Message pt. 1 — configuration

```
val strategy = Strategy.Builder()
     .setDiscoveryMode(Strategy.DISCOVERY_MODE_BROADCAST)
     .setTtlSeconds(Strategy.TTL_SECONDS_MAX)
     .build()

val publishOpts = PublishOptions.Builder()
     .setStrategy(strategy)
     .setCallback(object : PublishCallback() {
         override fun onExpired() {
             Log.d(TAG, "onExpired")
         }
     })
     .build()
```

Publish a Nearby Message pt. 2 — publish

```
val client = Nearby.getMessagesClient(this)
val message = Message("Hello, Firebase Thing!")

client.publish(message, publishOpts)
    .addOnSuccessListener(this) {
        Log.e(TAG, "publish success")
    }
    .addOnFailureListener(this) { e ->
        Log.e(TAG, "publish failed", e)
    }
```

Subscribe to a Nearby Message pt. 1 — configuration

```
val strategy = Strategy.Builder()
        .setDiscoveryMode(Strategy.DISCOVERY_MODE_SCAN)
        .setTtlSeconds(Strategy.TTL_SECONDS_MAX)
        .build()

val subscribeOpts = SubscribeOptions.Builder()
        .setStrategy(strategy)
        .setCallback(object : SubscribeCallback() {
            override fun onExpired() {
                Log.d(TAG, "onExpired")
            }
        })
        .build()
```

Subscribe to a Nearby Message pt. 2 — subscribe

```
val client = Nearby.getMessagesClient(this)

private val messageListener = object : MessageListener() {
    override fun onFound(message: Message) {
        // messsage.content contains payload
    }
    override fun onLost(message: Message) {}
}

client.subscribe(messageListener, subscribeOpts)
    .addOnSuccessListener(this) {
        Log.d(TAG, "subscribe success")
    }
    .addOnFailureListener(this) { e ->
        Log.e(TAG, "subscribe failure", e)
    }
```

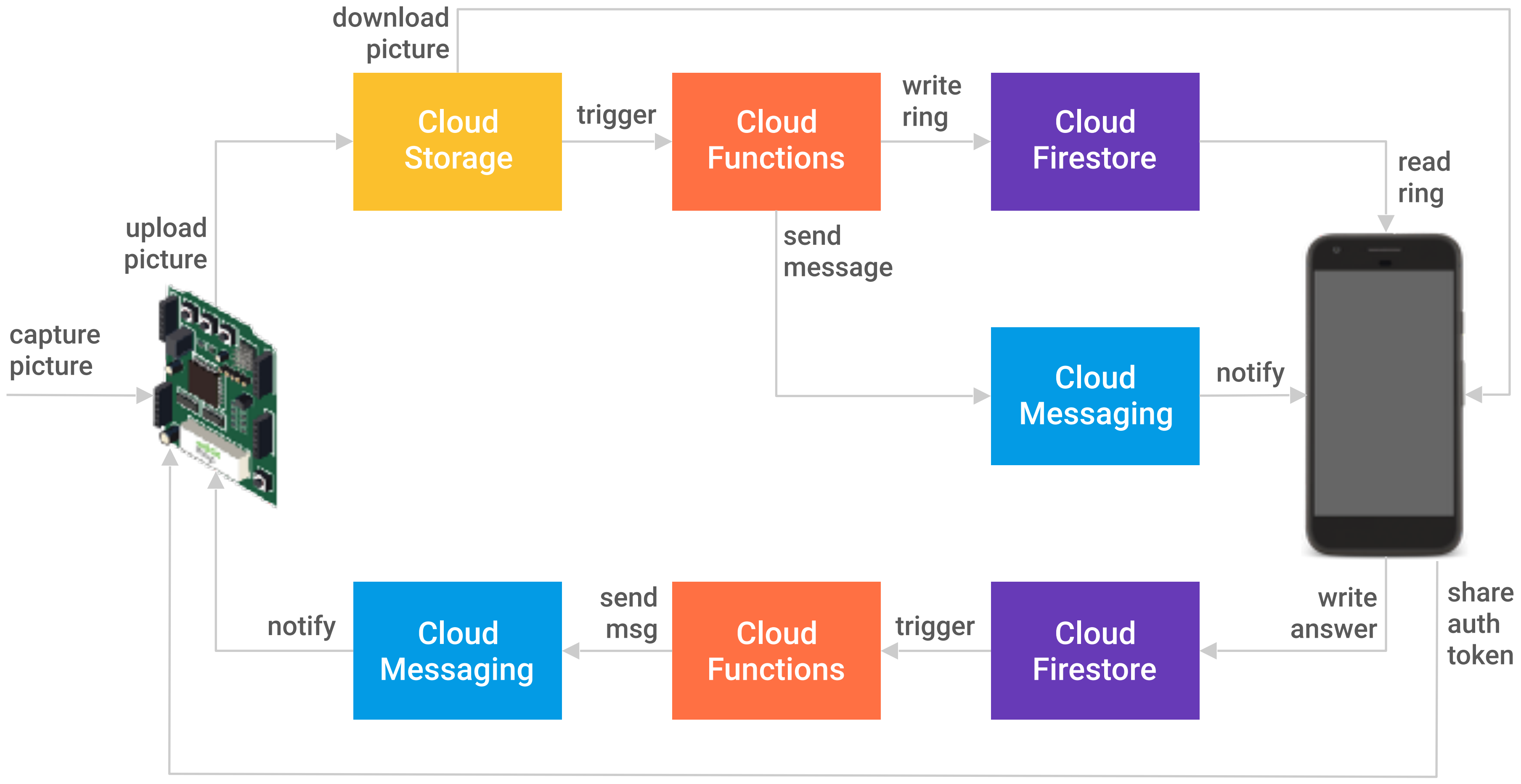
Log in Thing with Google and Firebase APIs

```
private fun trySignIn() {
    Log.d(TAG, "Signing in with token " + token)
    val credential = GoogleAuthProvider.getCredential(token, null)
    FirebaseAuth.getInstance().signInWithCredential(credential)
        .addOnSuccessListener(this, { result ->
            val user = result.user
            Log.d(TAG, "signInWithCredential ${user.displayName} ${user.email}")
            finish()
        })
        .addOnFailureListener(this, { e ->
            Log.e(TAG, "signInWithCredential onFailure", e)
        })
}
```

Use Nearby Connections to share a token (or anything)

Too much code to show here!

1. **Both**: Check for `ACCESS_COARSE_LOCATION` permission
2. **Thing**: Begin “advertising” with `P2P_CLUSTER` strategy
3. **App**: Begin “discovering” with `P2P_CLUSTER` strategy
4. **Both**: Connect to peer
5. **Both**: Send/receive Google Auth token
6. **Both**: Disconnect
7. **Thing**: Sign in with token using Google Auth and Firebase Auth APIs


download picture
upload picture
capture picture
Cloud Storage
trigger
Cloud Functions
write ring
Cloud Firestore
read ring
send message
Cloud Messaging
notify
notify
Cloud Messaging
send msg
Cloud Functions
trigger
Cloud Firestore
write answer
share auth token

@CodingDoug

What can be improved?

This project needs...

- Code cleanup
- Better UI
- Productization - currently only good for hobbyists
  - Can't expect a typical customer to manage a Firebase project
  - Currently no way to programmatically create a project
  - Restructure Firestore and Storage for multi-customer tenancy
  - Can’t use FCM topics securely - need to use device tokens

This project needs...

- Better security
    - Allow users to confirm auth tokens received from Nearby (like BT pairing)
    - Tighter security rules for both Storage and Firestore

What features can be added?

Some ideas

- Facial sentiment detection via Cloud Functions
- Voice intercom (AudioTrack, AudioRecord)
  - Add speech to text with Google Cloud Speech API
- “Digital keys”
  - Guest app
  - QR codes

Thank you!

@CodingDoug