PhMeter for Wine Making

Αθανάσιος Παληγιάννης 427
Σωτήριος Ζώγος 422

Τμήμα Μηχανικών Η/Υ και Πληροφορικής

Ενσωματωμένα Συστήματα και Εφαρμογές IoT
Τεχνική Αναφορά

Πανεπιστήμιο Ιωαννίνων

2/2/2020

Περιεχόμενα

1.1 Εισαγωγή 2

1.2 Ανάλυση σε MatLab 2

1.3. Ανάλυση σε Android Studio 4

2.1 Υλοποίηση συστήματος μέτρησης ph ενός διαλύμματος και εκτύπωση των αποτελεσμάτων στον φυλλομετρητή. 10

2.2 Συνοπτική περιγραφή της αρχιτεκτονικής της εφαρμογής μέτρησης Ph. 12

2.2.1 Αρχιτεκτονική ροής της εφαρμογής 13

2.2.2 Ανάλυση των επιμέρους τμημάτων της εφαρμογής 13

2.2.3 Ανάλυση του τμήματος κώδικα που τρέχει στον microcontroller 14

2.2.4 Ανάλυση του τμήματος κώδικα που τρέχει στον web server 18

2.2.4.1 Δομή του web server 19

2.2.4.2 Εισαγωγή του web server στο περιβάλλον ανάπτυξης Eclipse IDE for Java EE 21

2.2.4.3 Τμήματα της εφαρμογής που τρέχουν στον web server 21

2.2.4.4 JPA Hibernate API 21

2.2.4.5 Business logic 23

2.2.4.6 Restful API 23

2.2.5 Ανάλυση του τμήματος κώδικα που εκτελείται στην βάση δεδομένων 24

2.2.6 Ανάλυση του τμήματος κώδικα που εκτελείται στο front-end 25

3.1 Εισαγωγή 27

3.2 Υπολογισμός του resolution range. 27

3.3 Υπολογισμός του datarate range. 27

3.4 Υπολογισμός του ADC(Analog to Digital Conversion) power. 27

3.5 Εισαγωγή των δεδομένων στην Android εφαρμογή μας. 27

Μέρος πρώτο

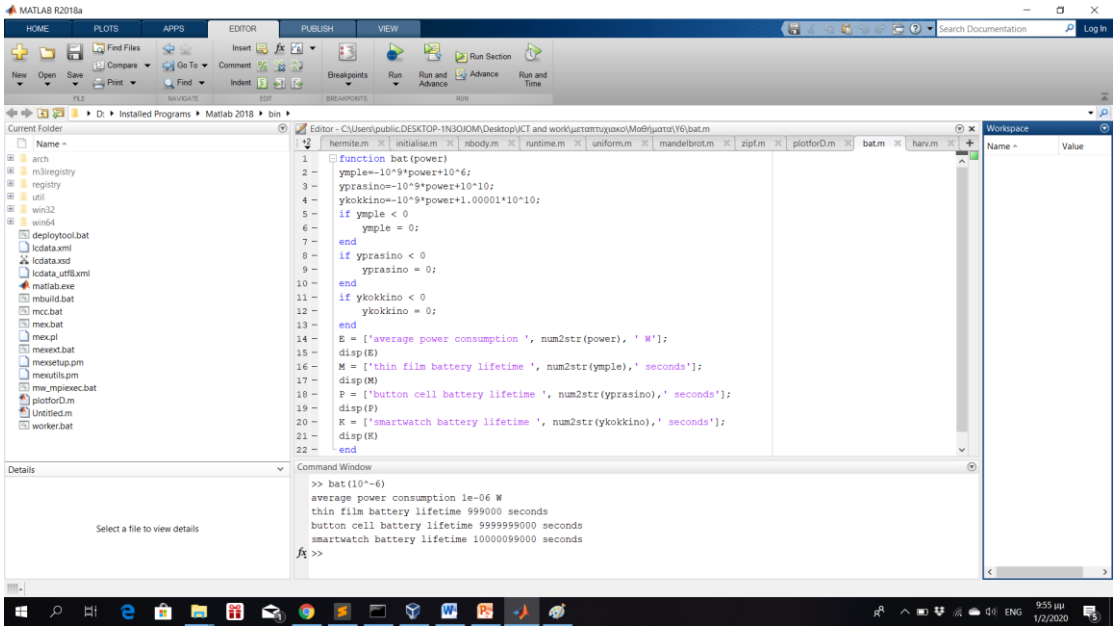
1.1 Εισαγωγή

Το πρώτο μέρος της εργασίας μας περιλαμβάνει την μοντελοποίηση των πέντε διαγραμμάτων που υπήρχαν στο πρώτο κεφάλαιο του βιβλίου “Enabling the internet of things” του “Massimo Alioto”. Τα διαγράμματα που μοντελοποιήθηκαν χωρίστηκαν σε δύο κατηγορίες. Η πρώτη κατηγορία περιλαμβάνει δυο διαγράμματα που αναφέρονται στο βιβλίο ως “Lifetime vs average power consumption for different batteries”. Το πρώτο διάγραμμα απεικονίζει την διάρκεια ζωής τριών τύπων μπαταριών (smartwatch battery, button cell battery, thin-film battery) σε σχέση με την μέση κατανάλωση ενέργειας για κάθε μπαταρία. Το δεύτερο διάγραμμα απεικονίζει το αναμενόμενο μέγεθος του καταχωρητή (harvester) προς την αναμενόμενη κατανάλωση ενέργειας από την iot συσκευή. Στο δεύτερο διάγραμμα υπάρχουν οι εξής τύποι καταχωρητών: photovoltaic (indoor), thermal (human), vibration/motion (human) , electromagnetic (GSM), photovoltaic (indoor), thermal (industrial), air flow (indoor). Η μοντελοποίηση της πρώτης κατηγορίας διαγραμμάτων έγινε στο προγραμματιστικό πακέτο MatLab (όπως θα δείτε αναλυτικά παρακάτω). Η δεύτερη κατηγορία περιλαμβάνει τρία διαγράμματα. Το πρώτο διάγραμμα απεικονίζει το resolution range που απαιτείται για διάφορους αισθητήρες. Οι αισθητήρες που υπάρχουν στα διαγράμματα είναι οι ακόλουθοι: heart rate, humidity, battery monitor, temperature, accelerometer, magnetometer, altimeter/pressure, imager (VGA, RGB), imager (MP4 compressed), infrared proximity, gyroscope, microphone, CO2, light, strain, ultra-violet. Το δεύτερο διάγραμμα απεικονίζει το datarate range που απαιτείται από κάθε έναν αισθητήρα που προαναφέραμε καθώς και το wireless power που απαιτείται για να μεταδίδει συνεχόμενα τα δεδομένα των αισθητήρων η iot συσκευή που τους χρησιμοποιεί. Το τρίτο και τελευταίο διάγραμμα απεικονίζει την ενέργεια που χρειάζεται για την Analog to Digital μετατροπή των δεδομένων για κάθε έναν από τους παραπάνω αισθητήρες. Η δεύτερη κατηγορία διαγραμμάτων μοντελοποιήθηκε χρησιμοποιώντας το προγραμματιστικό περιβάλλον Android Studio (όπως θα αναλύσουμε στο κομμάτι 1.3).

1.2 Ανάλυση σε MatLab

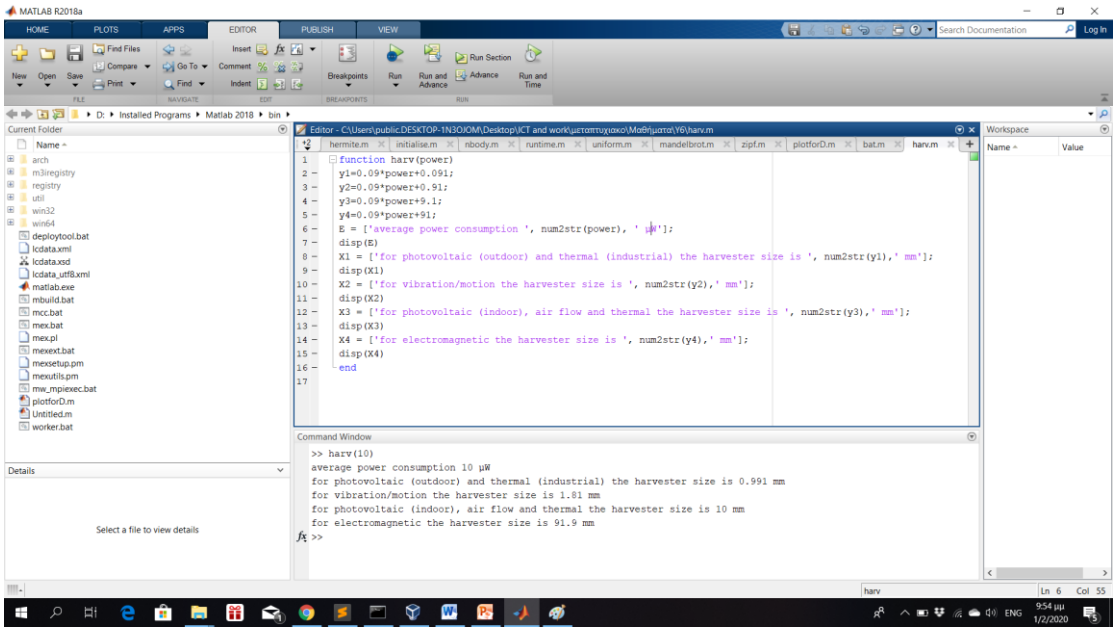
Όπως προαναφέραμε για την μοντελοποίηση των δύο μοντέλων της πρώτης κατηγορίας που αφορούσαν την διάρκεια ζωής των μπαταριών σε σχέση με την μέση κατανάλωση ισχύος της iot συσκευής χρησιμοποιήσαμε το προγραμματιστικό πακέτο MatLab. Αρχικά, από τα διαγράμματα του βιβλίου υπολογίσαμε με βάση της τιμές στους άξονες x και y τις εξισώσεις των τριών ευθειών που περιγράφανε την σχέση ανάμεσα στην κατανάλωση και την διάρκεια ζωής. Πιο αναλυτικά, γνωρίζοντας ότι οι τρεις αυτές σχέσεις θα είναι της μορφής $y = a * x + b$ καθώς ήταν γραμμικές (όπου στον άξονα y απεικονίζοταν η διάρκεια ζωής και στον άξονα x η μέση κατανάλωση ισχύος από την iot συσκευή), πήραμε δύο ζευγάρια τιμών (x,y) από την παράσταση και λύνοντας ένα 2x2 σύστημα εξισώσεων υπολογίσαμε τα a και b για κάθε μία εξίσωση ξεχωριστά. Ειδικότερα για την εξίσωση της μπαταρίας από smartwatch έχουμε $a = -10^9$ και $b = 10^6$, για τη εξίσωση της button cell μπαταρίας έχουμε $a = -10^9$ και $b = 10^{10}$, ενώ τέλος για την εξίσωση της thin-film μπαταρίας έχουμε $a = -10^9$ και $b = 1{,}00001 * 10^{10}$. Έτσι δημιουργήσαμε την συνάρτηση bat η οποία

δέχεται σαν όρισμα την μέση ισχύ (power) που καταναλώνει η iot συσκευή και επιστρέφει σαν αποτέλεσμα την διάρκεια ζωής κάθε μίας απ τις τρεις προαναφερθέντες μπαταρίες. Η δομή της συνάρτησης φαίνεται στην παρακάτω φωτογραφία.



Παράδειγμα εκτέλεσης της συνάρτησης bat με είσοδο power $= 10^{-6}W$.

Αντίστοιχα, την ίδια διαδικασία ακολουθήσαμε για την μοντελοποίηση του διαγράμματος για τους καταχωρητές (harvesters). Σ αυτό το διάγραμμα έιχαμε τέσσερις ευθείες καθώς κάποιοι τύποι ισχύος μοιράζονταν κάποια κοινή εξίσωση. Και σε αυτήν την περίπτωση είχαμε να λύσουμε ένα 2χ2 σύστημα εξισώσεων της μορφής $y = a * x + b$ χρησιμοποιώντας από το σχήμα δύο σημεία για κάθε μία εξίσωση. Σε όλες τις εξισώσεις παρατηρήσαμε ότι το $a = 0{,}09$ ενώ το b αλλάζει ανάλογα με τον τύπο των harvesters. Συγκεκριμένα για τους harvesters τύπου outdoor photovoltaic και τύπου industrial thermal υπολογίσαμε ότι $b = 0{,}091$, για τους harvesters τύπου vibration/motion έχουμε $b = 0{,}91$, για τους harvesters τύπου indoor photovoltaic , air flow και human thermal έχουμε $b = 9{,}1$, και τέλος για τον electromagnetic harvester το $b = 91$. Έτσι δημιουργήσαμε την συνάρτηση harv η οποία δέχεται ως είσοδο την τιμή της ισχύος σε (μW) και επιστρέφει το αναμενόμενο μέγεθος του harvester που θα αντικαθιστούσε μία κοινή button cell μπαταρία. Η δομή αυτής της συνάρτησης φαίνεται στην παρακάτω φωτογραφία.



Παράδειγμα εκτέλεσης της συνάρτησης bat με είσοδο power = 10μW.

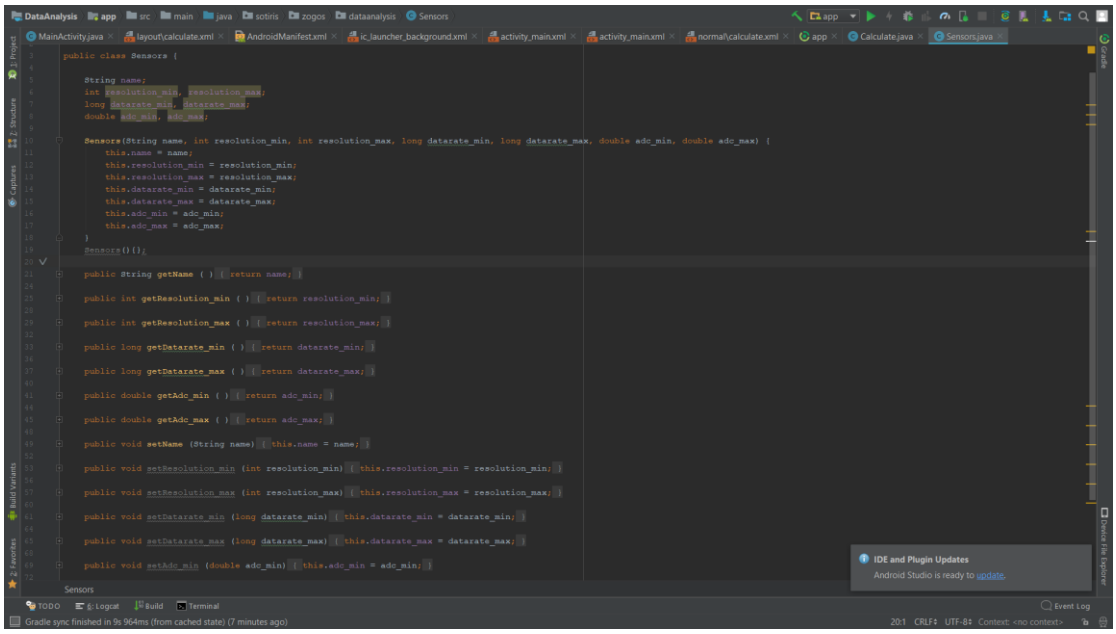
1.3. Ανάλυση σε Android Studio

Η δεύτερη κατηγορία διαγραμμάτων μοντελοποιήθηκε με την δημιουργία μιας android εφαρμογής για κινητά και tablet. Σκοπός της εφαρμογής αυτής ήταν να υπολογίζει το συνολικό resolution range (σε bits), το συνολικό datarate range (σε bps), την κατανάλωση ισχύος για την Analog to Digital μετατροπή των δεδομένων, όπως επίσης και την συνολική κατανάλωση ισχύος της iot συσκευής η οποία χρησιμοποιούσε μια ομάδα από αισθητήρες.

Για τον σκοπό αυτό ξεκινήσαμε δημιουργώντας μία κλάση “Sensors” η οποία περιέχει όλα τα δεδομένα που αντλήσαμε από τα τρία αυτά διαγράμματα του βιβλίου. Έτσι κάθε ένα αντικείμενο της κλάσης Sensors περιέχει 7 χαρακτηριστικά (attributes):

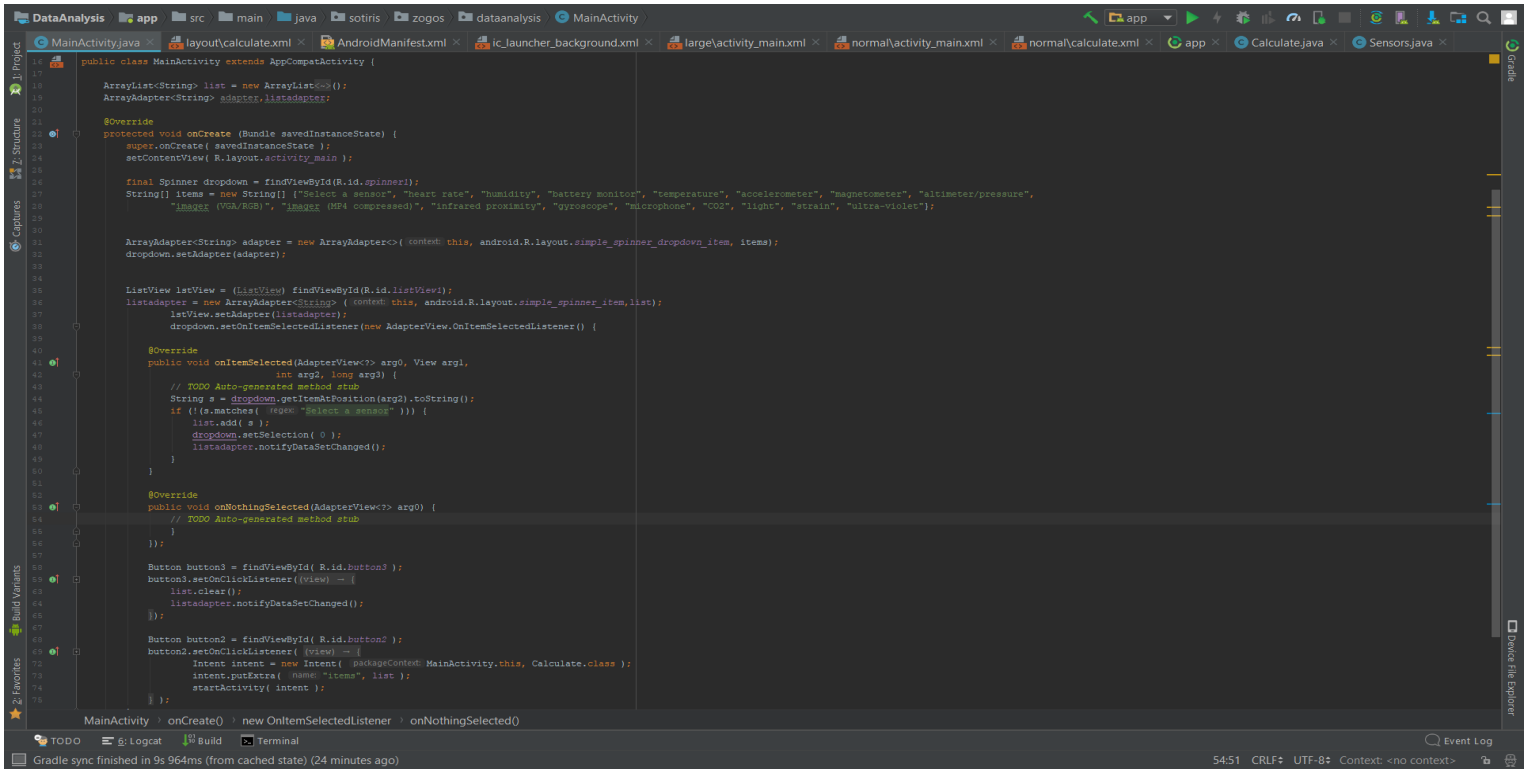
1. το όνομα του αισθητήρα (name): τύπου String
2. την ελάχιστη τιμή του resolution (resolution_min): τύπου int
3. την μέγιστη τιμή του resolution (resolution_max): τύπου int
4. την ελάχιστη τιμή του datarate (datarate_min): τύπου long
5. την μέγιστη τιμή του datarate (datarate_max): τύπου long
6. την ελάχιστη τιμή της Analog to Digital convertion (adc_min): τύπου double
7. την μέγιστη τιμή της Analog to Digital convertion (adc_max): τύπου double

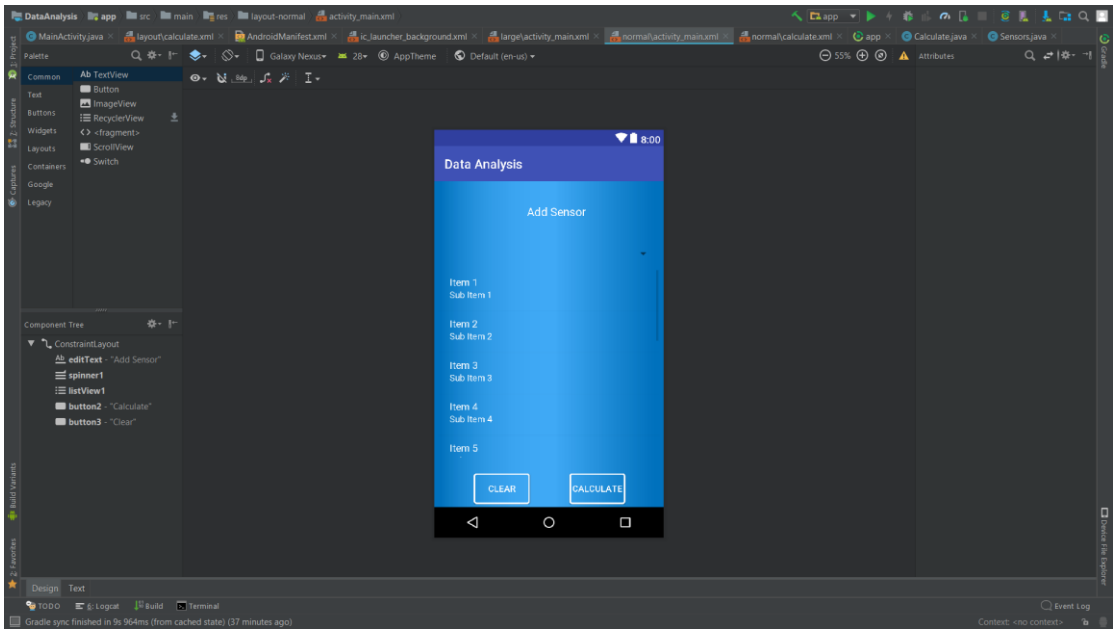
Η κλάση “Sensors” δεν περιέχει τίποτα άλλο εκτός από τους Constructors (ένας default, κι ένας με τα 7 attributes που προαναφέραμε) καθώς και τους αντίστοιχους getters και setters για κάθε ένα από τα attributes.



Η κλάση "Sensors" μέσα από το Android Studio.

Έπειτα δημιουργήσαμε δύο ακόμη κλάσεις, τις MainActivity και Calculate και δύο xml layouts, τα activity_main.xml και calculate.xml (ένα για κάθε κλάση αντίστοιχα).
Σκοπός της κλάσης MainActivity είναι να δημιουργήσει ένα περιβάλλον στο οποίο ο χρήστης θα μπορεί να επιλέξει τους αισθητήρες που επιθυμεί για την iot συσκευή του, μέσα από μια λίστα με όλους τους διαθέσιμους αισθητήρες. Αφού ο χρήστης επιλέξει τους αισθητήρες, πατώντας το κουμπί "calculate", η κλάση MainActivity αναλαμβάνει να περάσει την λίστα με τους επιλεγμένους αισθητήρες στην κλάση "Calculate" η οποία είναι υπεύθυνη για τον υπολογισμό του συνολικού μέγιστου και ελάχιστου resolution, του συνολικου μέγιστου και ελάχιστου datarate, του συνολικου μέγιστου και ελάχιστου adc power consumption καθώς και του μέγιστου και ελάχιστου total power consumption της iot συσκευής που θα χρησιμοποιεί αυτούς τους αισθητήρες.





MainActivity layout

Πιο αναλυτικά, στο παραπάνω κώδικα διακρίνουμε μία λίστα “items” η οποία περιέχει τα ονόματα των αισθητήρων που υπάρχουν στο dropdown menu που βλέπει ο χρήστης (όπως θα δουμε και παρακάτω), μια συνάρτηση onItemSelected που έχει ως σκοπό να προσθέτει τον αισθητήρα που επέλεξε ο χρήστης στην λίστα με τους επιλεγμένους αισθητήρες και να ανανεώνει την λίστα αυτή ώστε να εμφανίζεται ενημερωμένη στον χρήστη και δύο listeners για τα δύο κουμπιά (clear και calculate) που υπάρχουν στο layout που βλέπει ο χρήστης. Ο listener του κουμπιού “clear” έχει σκοπό να διαγράψει όλους του επιλεγόμενους αισθητήρες με σκοπό ο χρήστης να κάνει κάποιον νέο υπολογισμό, ενώ ο listener του κουμπιού “calculate” έχει σκοπό να περάσει στην κλάση “Calculate” με την χρήση του αντικειμένου τύπου Intent την λίστα με τους επιλεγμένους αισθητήρες έτσι ώστε να προχωρήσουμε στον υπολογισμό των αποτελεσμάτων.

Στην συνέχεια υλοποιήσαμε την κλάση “Calculate” η οποία περιέχει όλα τα δεδομένα που αντλήσαμε από τα τρία διαγράμματα, δέχεται την λίστα με τους επιλεγμένους αισθητήρες από την κλάση “MainActivity” μέσω της συνάρτησης getIntent() και για κάθε ένα αντικείμενο Sensors σ’ αυτην την λίστα χρησιμοποιώντας μια επανάλληψη for υπολογίζουμε τα resolution_min, resolution_max, datarate_min, datarate_max, adc_min, adc_max και total_min, total_max. Αφού γίνει ο υπολογισμός αυτός, τα αποτελέσματα εμφανίζονται στο layout που βλέπει ο χρήστης. Τέλος, ο χρήστης έχει μία τελευταία δυνατότητα, πατώντας το κουμπί “save to pdf”, τα αποτελέσματα αυτά με την βοήθεια ενός stringbuilder μετατρέπονται σε κείμενο και αποθηκεύονται σε μορφή pdf στην εσωτερική μνήμη της συσκευής.

```
import ...

public class Calculate extends AppCompatActivity {

    ArrayList<Sensors> sensors = new ArrayList();
    int resolution_min=0, resolution_max=0;
    long datarate_min=0, datarate_max=0;
    double adc_min=0, adc_max=0;
    double total_min=0, total_max=0;


    @Override
    protected void onCreate (Bundle savedInstanceState) {

        super.onCreate( savedInstanceState );
        setContentView( R.layout.calculate );

        Intent intent = getIntent();

        final ArrayList<String> list = getIntent().getStringArrayListExtra( name: "items");


        sensors.add( new Sensors( name: "heart rate",  resolution_min: 4,  resolution_max: 12,  datarate_min: 5000,  datarate_max: 10000,  adc_min: 0.000000001,  adc_max: 0.000000005 ) );
        sensors.add( new Sensors( name: "humidity",  resolution_min: 8,  resolution_max: 16,  datarate_min: 10,  datarate_max: 50,  adc_min: 0,  adc_max: 0 ) );
        sensors.add( new Sensors( name: "battery monitor",  resolution_min: 5,  resolution_max: 8,  datarate_min: 5,  datarate_max: 30,  adc_min: 0,  adc_max: 0 ) );
        sensors.add( new Sensors( name: "temperature",  resolution_min: 6,  resolution_max: 12,  datarate_min: 100,  datarate_max: 300,  adc_min: 0.0000000005,  adc_max: 0.000000001 ) );
        sensors.add( new Sensors( name: "accelerometer",  resolution_min: 8,  resolution_max: 14,  datarate_min: 2000,  datarate_max: 5000,  adc_min: 0.00000001,  adc_max: 0.00000003 ) );
        sensors.add( new Sensors( name: "magnetometer",  resolution_min: 8,  resolution_max: 16,  datarate_min: 800,  datarate_max: 1200,  adc_min: 0.000000007,  adc_max: 0.00000001 ) );
        sensors.add( new Sensors( name: "altimeter/pressure",  resolution_min: 8,  resolution_max: 24,  datarate_min: 800,  datarate_max: 2000,  adc_min: 0.000000007,  adc_max: 0.00000003 ) );
        sensors.add( new Sensors( name: "imager (VGA/BGR)",  resolution_min: 8,  resolution_max: 10,  datarate_min: 100000000,  datarate_max: 200000000,  adc_min: 0.0007,  adc_max: 0.001 ) );
        sensors.add( new Sensors( name: "imager (MP4 compressed)",  resolution_min: 8,  resolution_max: 10,  datarate_min: 10000000,  datarate_max: 300000000,  adc_min: 0.0007,  adc_max: 0.001 ) );
        sensors.add( new Sensors( name: "infrared proximity",  resolution_min: 10,  resolution_max: 16,  datarate_min: 300,  datarate_max: 800,  adc_min: 0.000000007,  adc_max: 0.000000013 ) );
        sensors.add( new Sensors( name: "gyroscope",  resolution_min: 12,  resolution_max: 16,  datarate_min: 10000,  datarate_max: 3000,  adc_min: 0.000002,  adc_max: 0.000008 ) );
        sensors.add( new Sensors( name: "microphone",  resolution_min: 12,  resolution_max: 16,  datarate_min: 200000,  datarate_max: 50000,  adc_min: 0.00002,  adc_max: 0.00008 ) );
        sensors.add( new Sensors( name: "CO2",  resolution_min: 14,  resolution_max: 16,  datarate_min: 300,  datarate_max: 700,  adc_min: 0.0000000004,  adc_max: 0.0000000006 ) );
        sensors.add( new Sensors( name: "light",  resolution_min: 16,  resolution_max: 24,  datarate_min: 200,  datarate_max: 500,  adc_min: 0.0000005,  adc_max: 0.0000008 ) );
        sensors.add( new Sensors( name: "strain",  resolution_min: 16,  resolution_max: 24,  datarate_min: 20000,  datarate_max: 50000,  adc_min: 0.00005,  adc_max: 0.00008 ) );
        sensors.add( new Sensors( name: "ultra-violet",  resolution_min: 0,  resolution_max: 0,  datarate_min: 400,  datarate_max: 700,  adc_min: 0.0000001,  adc_max: 0.0000002 ) );



        for (String s : list) {
```

Η αρχικοποίηση των αντικειμένων Sensors με τις τιμές από τα διαγράμματα του βιβλίου.

```
        for (String s : list) {
            for (Sensors sensor : sensors) {
                if (s.matches( sensor.getName() )){
                    resolution_min = resolution_min + sensor.getResolution_min();
                    resolution_max = resolution_max + sensor.getResolution_max();
                    datarate_min = datarate_min + sensor.getDatarate_min();
                    datarate_max = datarate_max + sensor.getDatarate_max();
                    adc_min = adc_min + sensor.getAdc_min();
                    adc_max = adc_max + sensor.getAdc_max();
                }
            }

        }

        total_min=adc_min+datarate_min*5*(pow(10,-9));
        total_max=adc_max+datarate_max*5*(pow(10,-9));

        TextView textView2 = findViewById( R.id.textView2 );
        textView2.setText( ""+resolution_max+" bits" );

        TextView textView6 = findViewById( R.id.textView6 );
        textView6.setText( resolution_min+" bits" );

        TextView textView8 = findViewById( R.id.textView8 );
        textView8.setText( datarate_min+" bps" );

        TextView textView10 = findViewById( R.id.textView10 );
        textView10.setText( datarate_max+" bps" );

        TextView textView12 = findViewById( R.id.textView12 );
        textView12.setText( adc_min+" W" );

        TextView textView14 = findViewById( R.id.textView14 );
        textView14.setText( adc_max+" W" );

        TextView textView20 = findViewById( R.id.textView20 );
        textView20.setText( total_min+" W" );

        TextView textView22 = findViewById( R.id.textView22 );
        textView22.setText( total_max+" W" );
```

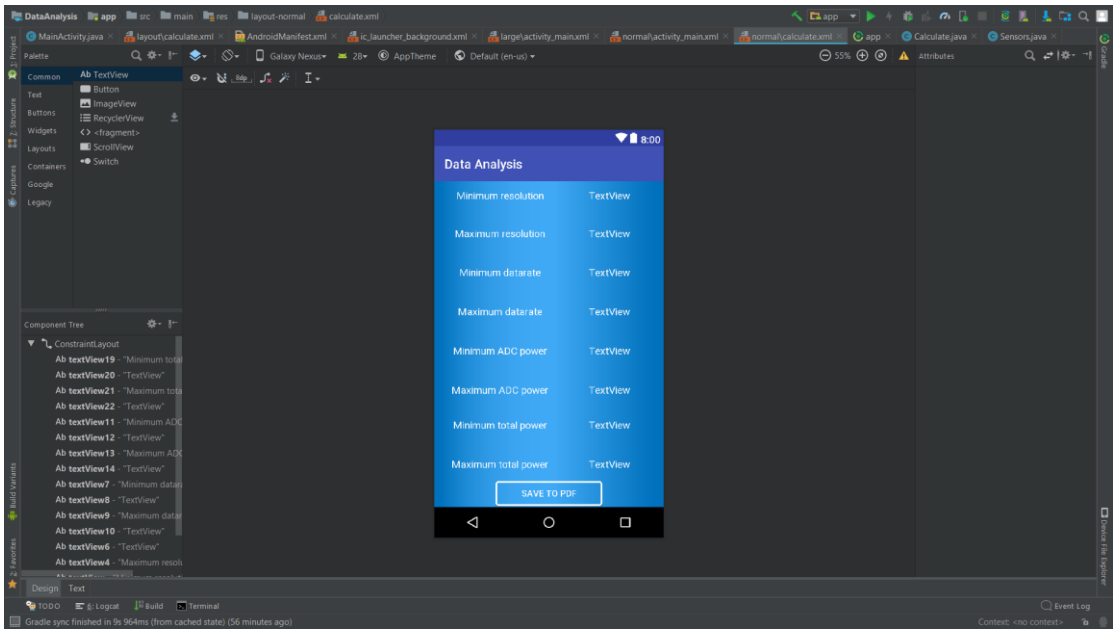
Η συνάρτηση υπολογισμού των αποτελεσμάτων και οι απεικονίσεις στο layout.

```
        TextView textView22 = findViewById( R.id.textView22 );
        textView22.setText( total_max+" W" );


        Button buttonpdf = findViewById( R.id.buttonpdf );
        buttonpdf.setOnClickListener( (view) -> {
                StringBuilder sb = new StringBuilder();
                sb.append( "These are the selected sensors:" );
                sb.append( "\n" );
                for (String s : list) {
                    sb.append( s );
                    sb.append( "\n" );
                }
                sb.append( "This is your analysis results" );
                sb.append( "\n" );
                sb.append( "Minimum resolution: " + resolution_min + " bits");
                sb.append( "\n" );
                sb.append( "Maximum resolution: " + resolution_max + " bits");
                sb.append( "\n" );
                sb.append( "Minimum datarate: " + datarate_min + " bps");
                sb.append( "\n" );
                sb.append( "Maximum datarate: " + datarate_max + " bps" );
                sb.append( "\n" );
                sb.append( "Minimum ADC power: " + adc_min + " W");
                sb.append( "\n" );
                sb.append( "Maximum ADC power: " + adc_max + " W");
                sb.append( "\n" );
                sb.append( "Minimum total power: " + total_min + " W");
                sb.append( "\n" );
                sb.append( "Maximum total power: " + total_max + " W");
                sb.append( "\n" );
                String sometext = sb.toString();
                createPdf( sometext );
        } );


        }

    private void createPdf(String sometext) {
        // create a new document
        PdfDocument document = new PdfDocument();
        // crate a page description
        PdfDocument.PageInfo pageInfo = new PdfDocument.PageInfo.Builder( pageWidth: 300,  pageHeight: 600,  pageNumber: 1 ).create();
```

Η μετατροπή των αποτελεσμάτων σε κείμενο.

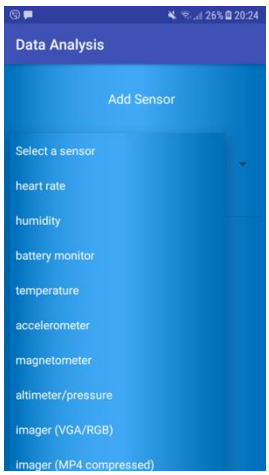
```
    }

    private void createPdf(String sometext) {
        // create a new document
        PdfDocument document = new PdfDocument();
        // crate a page description
        PdfDocument.PageInfo pageInfo = new PdfDocument.PageInfo.Builder( pageWidth: 300,  pageHeight: 600,  pageNumber: 1 ).create();
        // start a page
        PdfDocument.Page page = document.startPage( pageInfo );
        Canvas canvas = page.getCanvas();
        Paint paint = new Paint();
        paint.setColor( Color.RED );
        paint.setColor( Color.BLACK );
        paint.setTextSize( 11 );

        int y = 40;
        String[] lines = sometext.split( regex: "\n" );
        for (int i=0; i<lines.length; i++) {
            canvas.drawText(lines[i],  x: 20, y, paint);
            y+=20;
        }

        document.finishPage( page );
        File root = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMENTS);
        File file = new File(  pathname: root + "/Data Analysis" );
        if (!file.exists()) {
            file.mkdirs();
        }
        SimpleDateFormat df = new SimpleDateFormat( pattern: "dd-MM-yyyy");
        String formatted = df.format(new Date());

        String fname = "YourAnalysis-"+formatted+".pdf";
        File filePath = new File( file, fname );
        try {
            document.writeTo( new FileOutputStream( filePath ) );
            Toast.makeText(  context: this,  text: "Done", Toast.LENGTH_LONG ).show();
        } catch (IOException e) {
            Log.e(  tag: "main",  msg: "error " + e.toString() );
            Toast.makeText(  context: this,  text: "Something wrong: " + e.toString(), Toast.LENGTH_LONG ).show();
        }
        document.close();
    }
}
```

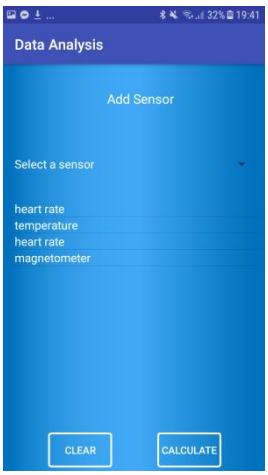
Η συνάρτηση createPdf η οποία δέχεται ως είσοδο το κείμενο, το μετατρέπει σε κατάλληλο format και το αποθηκεύει στην συσκευή σε μορφή pdf.



Το layout που βλέπει ο χρήστης

Παρακάτω παραθέτουμε ένα ολοκληρωμένο παράδειγμα χρήσης της εφαρμογής αυτής.





Εικόνα 1 *Εικόνα 2* *Εικόνα 3* *Εικόνα 4*

Ο χρήστης στην Εικόνα 1 έχει την δυνατότητα να επιλέξει τους αισθητήρες που επιθυμεί ανάμεσα σε μια λίστα από διαθέσιμους αισθητήρες. Έπειτα στην Εικόνα 2, το σύστημα εμφανίζει στον χρήστη τους αισθητήρες που επέλεξε καθώς και τις επιλογές για “clear” ή “calculate”. Πατώντας το κουμπί “calculate”, ο χρήστης μεταφέρεται στην Εικόνα 3, στην οποία το πρόγραμμα του παρουσιάζει τα αποτελέσματα για την iot συσκευή που έχει τους αισθητήρες που επέλεξε. Τελος πατώντας το κουμπί “save to pdf”, παράγεται και αποθηκεύεται στην συσκευή ένα αρχείο τύπου pdf που έχει το format της Εικόνας 4.

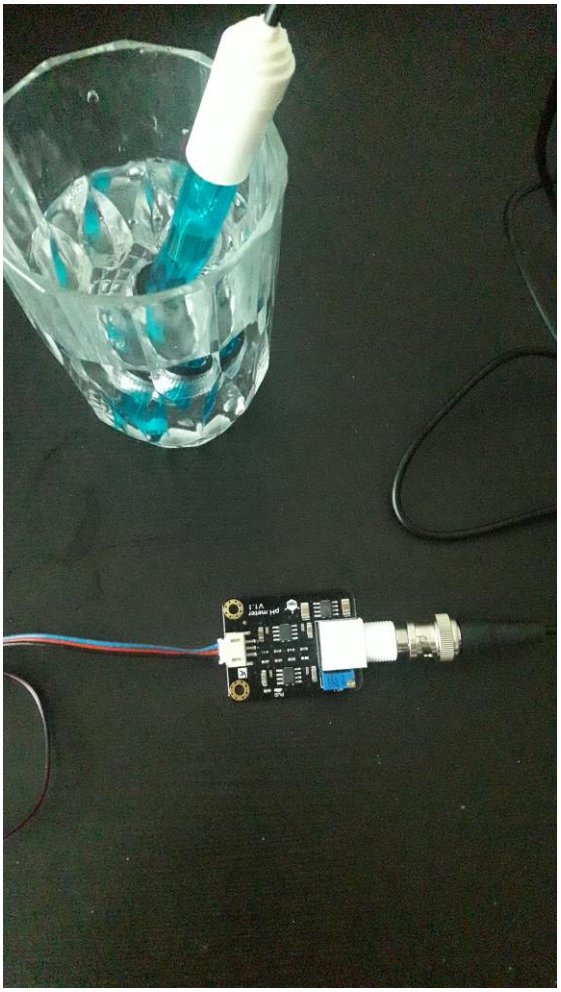
Μέρος 2ο

2.1 Υλοποίηση συστήματος μέτρησης ph ενός διαλύμματος και εκτύπωση των αποτελεσμάτων στον φυλλομετρητή.

Για την παραπάνω υλοποίηση χρησιμοποιήθηκαν τα παρακάτω εξαρτήματα

- Analog Ph Sensor for Arduino
- FRDM - K64F microcontroller
- Apache Tomee Server
- MySQL database
- Node.js Server
- Google Chrome

Αρχικά θα συνδέσουμε τον αισθητήρα μας με ένα κύκλωμα το οποίο δίνει τρείς εξόδους που θα αποτελέσουν είσοδοι για τον μικροεπεξεργαστή μας.



Όπως βλέπουμε και απο την εικόνα υπάρχουν 3 έξοδοι. Έχουμε 1 μαύρο καλώδιο, 1 κόκκινο καλώδιο και 1 μπλε καλώδιο.

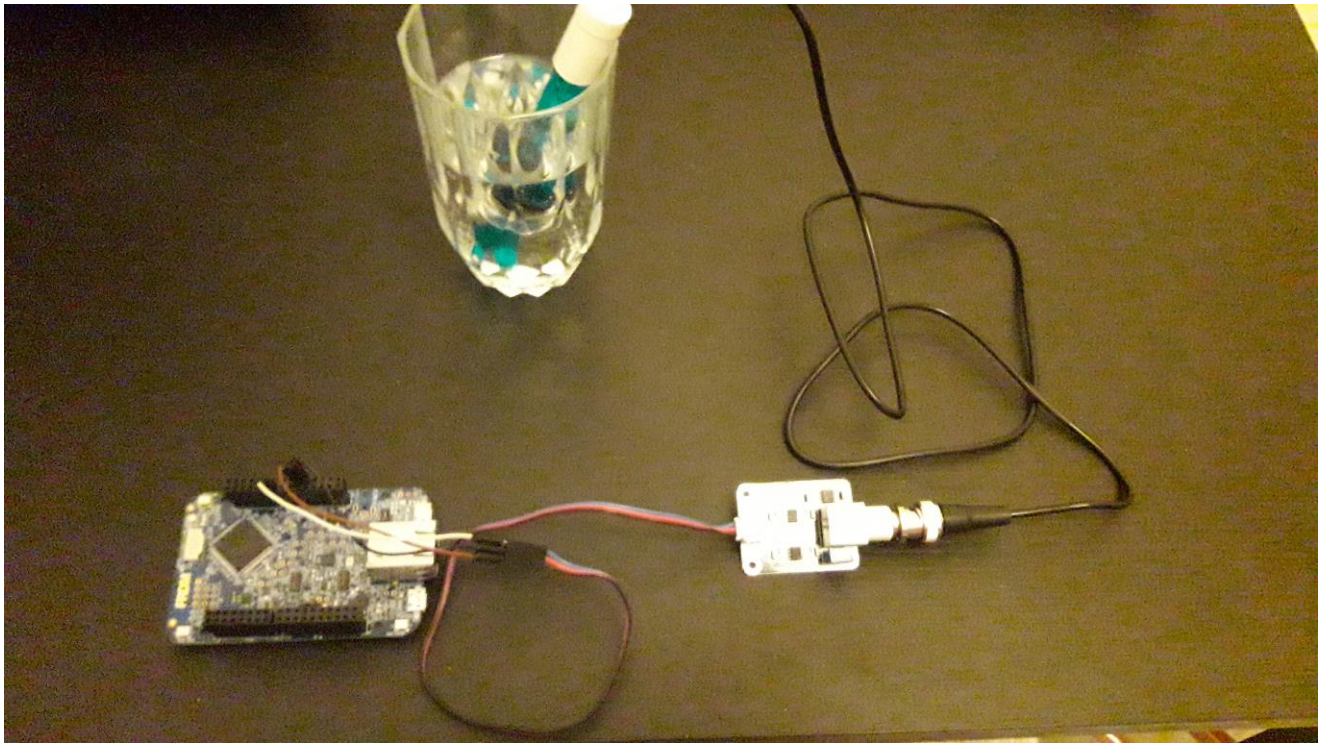
Μαύρο καλώδιο -> είναι η γείωση

Κόκκινο καλώδιο -> είναι η τάση στο κύκλώμα (5 Volt)

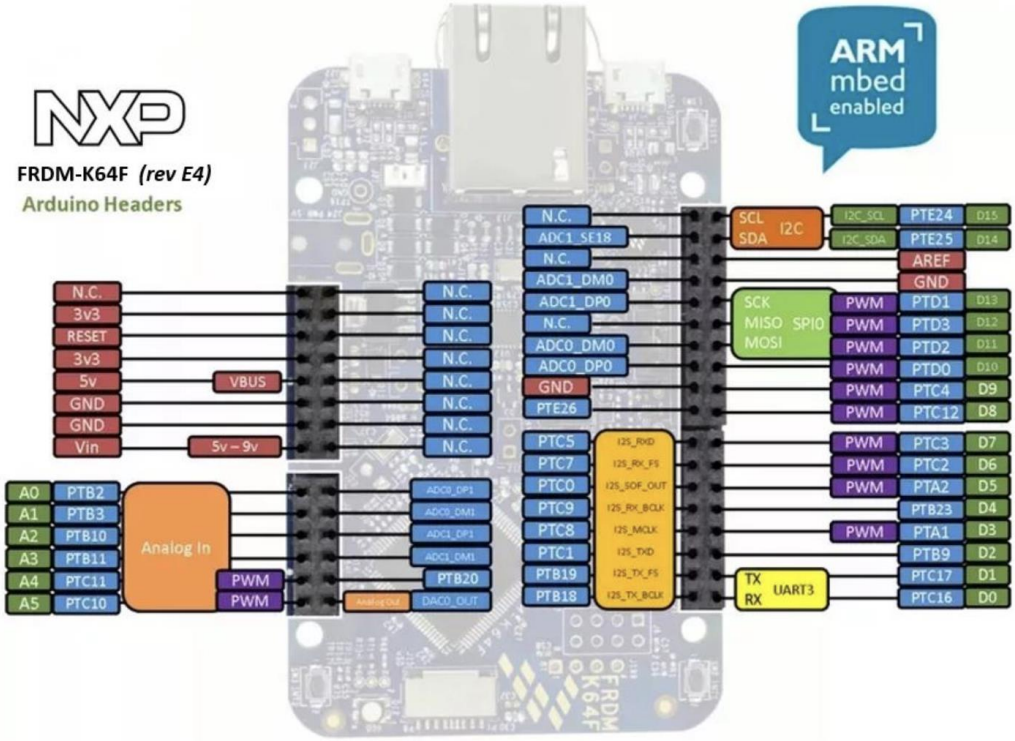
Μπλε καλώδιο -> είναι τα δεδομένα που δίνει ο αισθητήρας.

Ο συγκεκριμένος αισθητήρας λειτουργεί με τάση των 5 Volt. Ο συγκεκριμένος αισθητήρας είχε bnc αρσενικό βύσμα συνεπώς για να είναι συμβατός βρήκαμε το κύκλωμα στην εικόνα που είχε θηλυκό bnc βύσμα για να ταιριάξουν κατάλληλα.

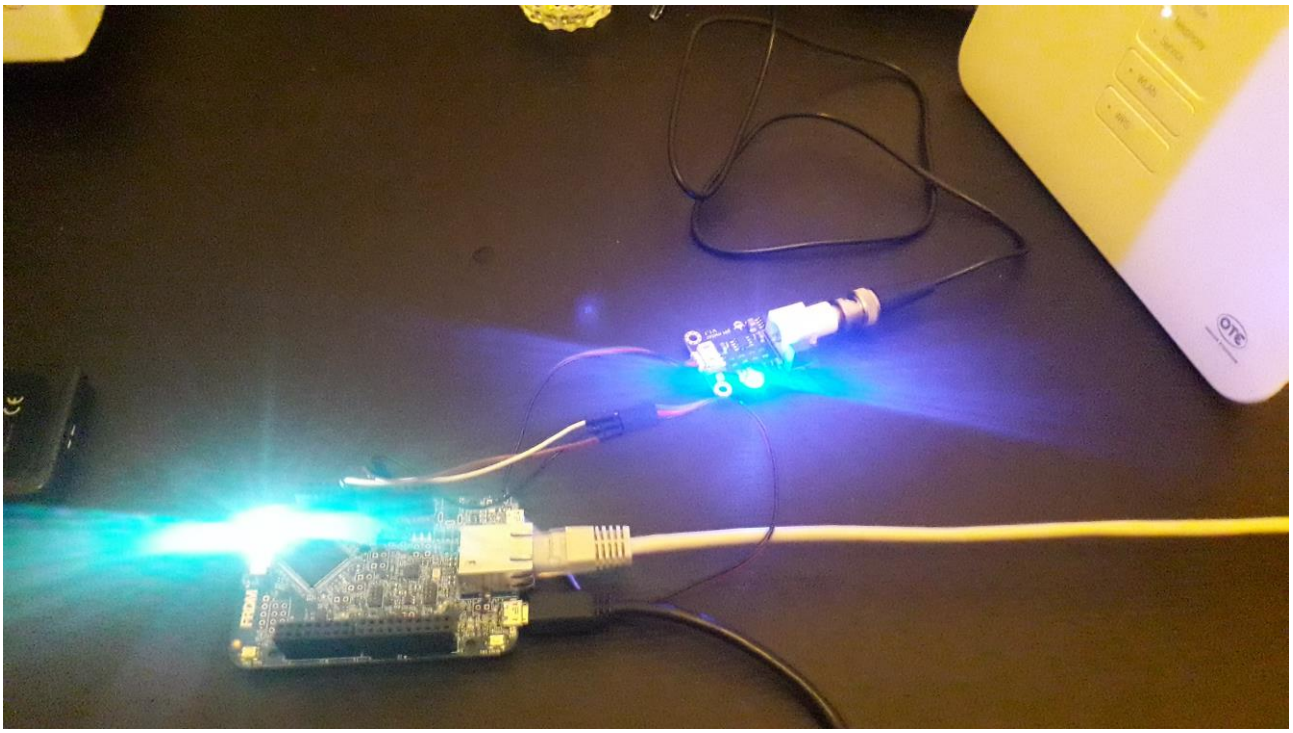
Στην συνέχεια θα συνδέσουμε το παραπάνω κύκλωμα του αισθητήρα με τον μικροεπεξεργαστή μας ο οποίος είναι ο FRDM-K64F. Στην επιλογή του συγκεκριμένου αισθητήρα έπαιξε ρόλο και η υποστήριξη σύνδεσης στο δίκτυο μέσω ethernet.



Με χρήση των καλωδίων “jumpers” συνδέσαμε το κύκλωμα του μετατροπέα με του μικροεπεξεργαστή. Σε αυτό το σημείο πρέπει να δώσουμε ιδιαίτερη προσοχή στο πως θα γίνει η μεταξύ τους σύνδεση. Για αυτόν τον λόγο ανατρέξαμε στο “https://os.mbed.com/platforms/FRDM-K64F/" που περιλαμβάνει οτιδήποτε αφορά τον μικροεπεξεργαστή μας.



Τώρα είμαστε έτοιμοι να συνδέσουμε τον μικροεπεξεργαστή μας με το δίκτυο. Για αύτον τον λόγο θα χρησιμοποιήσουμε ένα δίκτυο τύπου ethernet όπου θα συνδέσουμε τον μικροεπεξεργαστή μας με τον δρομολογητή του τοπικού δικτύου μας.



Συνεπώς τώρα ότι αφορά την συνδεσμολογία μας για την εφαρμογή έχει τελειώσει. Στην συνέχεια αυτό που κάναμε ήταν να προγραμματίσουμε τον μικροεπεξεργαστή μας να λαμβάνει τα δεδομένα απο την A0 (είναι η θύρα που έχουμε συνδέσει και είναι αυτήν που δέχεται αναλογικό input) και περιοδικά να στέλνει στον τοπικό εξυπηρετητή μας τα δεδομένα. Ο μικροεπεξεργαστής FRDM - K64F υποστηρίζει την πλατφόρμα mbed για αυτό και χρησιμοποιήσαμε τον online mbed compiler που προσφέρεται δωρεάν απο την ARM και μας παράγει τα εκτελέσιμα αρχεία .bin τα οποία φορτώνουμε στον μικροεπεξεργαστή μας.

2.2 Συνοπτική περιγραφή της αρχιτεκτονικής της εφαρμογής μέτρησης Ph.

Πίνακας Περιεχομένων:

-Αρχιτεκτονική ροής της εφαρμογής

-Ανάλυση των επιμέρους τμημάτων της εφαρμογής

-Ανάλυση του τμήματος κώδικα που τρέχει στον microcontroller

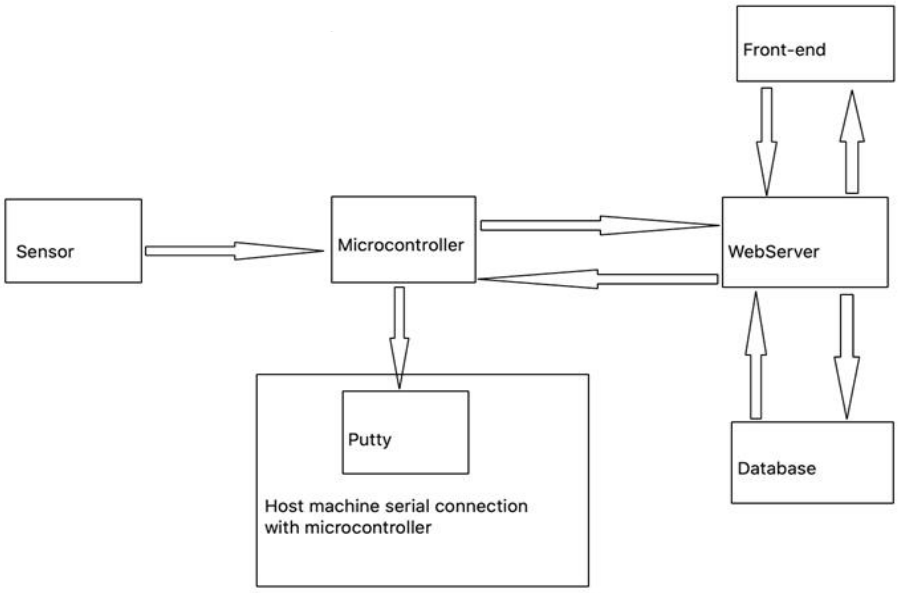
-Ανάλυση του τμήματος κώδικα που τρέχει στον web server

-Ανάλυση του τμήματος κώδικα που τρέχει στην βάση δεδομένων

-Ανάλυση του τμήματος κώδικα που τρέχει στο front-end

2.2.1 Αρχιτεκτονική ροής της εφαρμογής

Αρχιτεκτονική ροής της εφαρμογής φαίνεται στο παρακάτω σχήμα.



2.2.2 Ανάλυση των επιμέρους τμημάτων της εφαρμογής

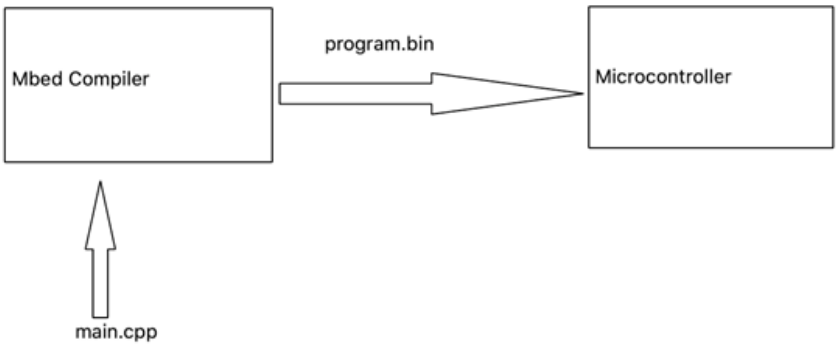
Όπως βλέπουμε στο παραπάνω σχήμα στην εφαρμογή μας συμμετέχουν διάφορες οντότητες όπου κάθεμια την έχουμε συμβολίσει ως ένα ορθογώνιο παραλληλόγραμμο. Αρχικά ο sensor δίνει την πληροφορία - δεδομένο στον microcontroller. Στην συνέχεια ο microcontroller εκτελεί δυο ενέργειες ανά 10 δευτερόλεπτα. Η πρώτη ενέργεια είναι να στέλνει περιοδικά μέσω δικτύου το δεδομένο που λαμβάνει απο τον sensor στον web server και η δεύτερη ενέργεια είναι να λαμβάνει περιοδικά απο τον web server τυχόν ενέργειες και να τις στέλνει σειριακά στο μηχάνημα που είναι συνδεδεμένος και να τις εκτυπώνει στο putty.

Όσον αφορά την λειτουργία του web server είναι να λαμβάνει απο τον microcontroller δεδομένα και να αποθηκεύει στην βάση δεδομένων που είναι συνδεδεμένη μαζί του. Επίσης στέλνει τυχόν ενέργειες που λαμβάνει απο το front-end στον microcontroller αφού τις έχει αποθηκεύσει πρώτα στην βάση δεδομένων.

Η βάση δεδομένων είναι υπεύθυνη για την αποθήκευση των δεδομένων που έρχονται απο τον microcontroller καθώς και για την απόθηκευση της ενέργειας που επιλέγει ο χρήστης.

Στο front-end κομμάτι ο χρήστης μπορεί να δεί τις τιμές του ph του διαλύμματος καθώς και να επιλέξει μία ενέργεια η οποία θα φτάσει εώς τον microcontroller.

2.2.3 Ανάλυση του τμήματος κώδικα που τρέχει στον microcontroller



Επειδή ο microcontroller που χρησιμοποιήσαμε ήταν ο FRDM - K64F όπου χρησιμοποιεί την πλατφόρμα του mbed για αυτόν τον λόγο χρησιμοποίησαμε και εμείς τον online mbed compiler. Ο κώδικας που γράψαμε ήταν γραμμένος σε C++. Απο εκεί και πέρα όταν ο compiler τρέξει το πρόγραμμα μας δημιουργεί ένα εκτελέσιμο (.bin). Στην συνέχεια το συγκεκριμένο εκτελέσιμο θα πρέπει να το φορτώσουμε στον microcontroller. Βέβαια για να τρέξει ο microcontroller το εκτελέσιμο που φορτώσαμε θα πρέπει να πατήσουμε το κουμπί reset που διαθέτει. Αφού κάναμε περιγραφή της διαδικασίας είμαστε έτοιμοι να προχωρήσουμε στην ανάλυση του κώδικα.

main.cpp

```
/* mbed Microcontroller Library
 * Copyright (c) 2019 ARM Limited
 * SPDX-License-Identifier: Apache-2.0
 */

#include "mbed.h"
#include "http_request.h"
#include "mbed_mem_trace.h"
#include <iostream>


Timer timer1;
Timer timer2;

AnalogIn Ain(A0);
float ADCdata;
Serial pc(USBTX, USBRX);

```

Απο την παραπάνω εικόνα βλέπουμε την δήλωση μεταβλητών για το πρόγραμμα μας. Αρχικά επειδή θέλουμε να συμβαίνουν δύο περιοδικά φαινόμενα, ορίσαμε δύο αντικείμενα τύπου Timer. Κατα την διάρκεια υλοποίησης είχαμε επιλέξει να γίνει με την χρήση των Ticker objects αλλά δυστυχώς για κάποιον άγνωστο λόγο το πρόγραμμα μας έσκαγε και αυτό οφείλοταν στην χρήση των Ticker. Ενώ όταν επιλέξαμε να χρησιμοποιήσουμε τα αντικείμενα τύπου Timer παρατηρήσαμε ότι ο microcontroller εκτελόυσε με επιτυχία τις ενέργειες περιοδικά χωρίς να σκάει.

Επίσης επειδή ο sensor στέλνει αναλογικά δεδομένα χρησιμοποιούμε την αναλογική είσοδο του microcontroller. Έχουμε επιλέξει την θύρα A0 καθώς με αυτήν την θύρα έχουμε συνδέσει το καλώδιο του sensor που μεταφέρει τα δεδομένα. Επιπλέον θέλουμε να παρατηρούμε για τυχόν ενέργειες και να τις εκτυπώνουμε στο putty μέσω σειριακής σύνδεσης, για αυτόν τον λόγο έχουμε ορίσει και την μεταβλητή τύπου Serial.

Στην επόμενη εικόνα θα δούμε το σώμα της κύριας συνάρτησης που τρέχει στον microcontroller και θα αναφερθουμε εκτενώς στις λειτουργίες που εκτελούνται μέσα σε αυτήν.

main.cpp

```
121
122 int main()
123 {
124     timer1.start();
125     timer2.start();
126     NetworkInterface* network = connectToNet();
127     char* url = "http://192.168.1.3:8080/MeterOfPhRest/action";
128     char* url1 = "http://192.168.1.3:8080/MeterOfPhRest/phdata/add";
129     const float m = -18.75;
130     const float b = 15.53;
131
132     while (true) {
133
134         if(timer1.read() >= 10)
135         {
136             sendGetRequest(network,url);
137             timer1.reset();
138         }
139
140        if(timer2.read() >= 10)
141         {
142            ADCdata=Ain;
143            float ph = ADCdata * m + b;
144            sendPostRequest(network, url1, ph);
145            timer2.reset();
146         }
147     }
148 }
149
```

Το τμήμα που μας ενδιαφέρει περικλείεται μέσα στο while block. Είναι το τμήμα που θέλουμε να τρέχει συνέχεια. Έξω απο το while block κάνουμε εκκίνηση των timers και των μεταβλητών που θα χρειαστούν στα αιτήματα (requests) που θα κάνει ο microcontroller. Θα ήταν καταστροφικό να τα έχουμε βάλει όλα μεσα στο while block γιατι κάθε φορά θα αρχικοποιούταν αυτές οι μεταβλητές χωρίς να υπάρχει λόγος και ίσως να είχαμε και άλλες δυσμενείς συνέπειες. Επιλέξαμε 10 δευτερόλεπτα να είναι η περίοδος που ο microcontroller στέλνει τα δεδομένα στον web server καθώς και στην ίδια αυτή περίοδο ακούει για τυχόν ενέργειες. Στο πρώτο if-block κάνει αίτημα για να πάρει τυχόν ενέργειες απο τον web server ,αυτό γίνεται με την βοήθεια της συνάρτησης sendGetRequest και στο δεύτερο if-block στέλνει τα δεδομένα περιοδικά στον web server με την βοήθεια της συνάρτησης sendPostRequest.

Οι συγκεκριμένες συναρτήσεις που αναφέρθηκαν πιο πάνω (sendGetRequest, sendPostRequest) κάνουν http requests στον web server. Έτσι λοιπόν ανατρέξαμε στο http api για να δούμε πως υλοποιούνται τα συγκεριμένα requests. Αρχικά είναι απαραίτητη η δημιουργία ενός αντικειμένου τύπου Network Interface το οποίο χρησιμοποιείται στις παραπάνω συναρτήσεις και υποδηλώνει την δημιουργία του network connection.

“NetworkInterface* network = connectToNet();”

```
/// declare network interface
NetworkInterface* connectToNet()
{
    NetworkInterface *net = NetworkInterface::get_default_instance();
    pc.printf("Establish connection ...");

    if (!net) {
        pc.printf("Error! No network inteface found.\n");
        return 0;
    }

    nsapi_error_t connect_status = net->connect();

    if (connect_status != NSAPI_ERROR_OK) {
        pc.printf(" Failed to connect to network (%d)\n", connect_status);
        return NULL;
    }

    pc.printf("Connected to the network\n");

    return net;
}
```

Το αντικείμενο τύπου NetworkInterface είναι που χρησιμοποιείται σε κάθε request που πραγματοποιεί ο microcontroller.Υπάρχουν δύο είδη request που εκτελέι ο microcontroller.

1. Post Request
2. Get Request

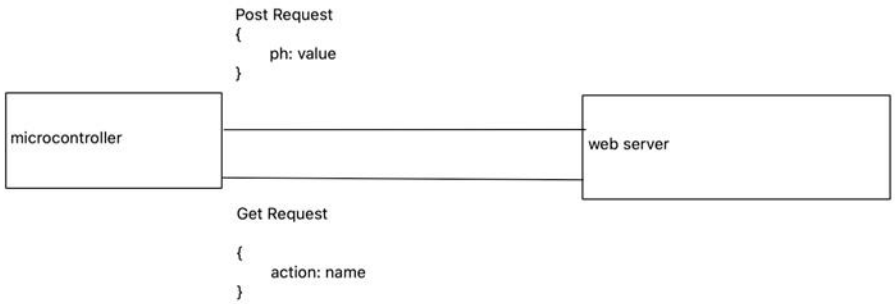
Με την χρήση του post request στέλνουμε τα δεδομένα στον web server ενώ με την χρήση get request λαμβάνουμε τα δεδομένα απο τον web server.Συνεπώς είμαστε έτοιμοι με την ανάλυση του σώματος των συναρτήσεων που κάνουν post and get request.

```
void sendPostRequest(NetworkInterface* network, char* url, float phNum)
    {
        HttpRequest* post_req = new HttpRequest(network, HTTP_POST, url);
         post_req->set_header("Content-Type", "application/json");
          char body[60] ;
         sprintf(body ,"{\"ph\": \"%.7f\" }", phNum);

         HttpResponse* post_res = post_req->send(body, strlen(body));
         if (!post_res) {
             printf("HttpRequest failed (error code %d)\n", post_req->get_error());
             return;
         }

         printf("\n----- HTTP POST response -----\n");
         receiveGetResponse(post_res);

         delete post_req;
    }

```

```
void sendGetRequest(NetworkInterface* network, char* url)
// Do a GET request
    {
        // By default the body is automatically parsed and stored in a buffer, this is memory heavy.
        // To receive chunked response, pass in a callback as last parameter to the constructor.

        HttpRequest* get_req = new HttpRequest(network, HTTP_GET, url);

        HttpResponse* get_res = get_req->send();
        if (!get_res) {
            pc.printf("HttpRequest failed (error code %d)\n", get_req->get_error());
            return ;
        }

        pc.printf("\n----- HTTP GET response -----\n");
        receiveGetResponse(get_res);

        delete get_req;
    }
```

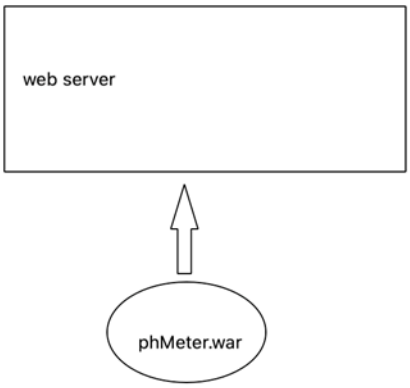
Μεταξύ του δικτύου τα δεδομένα που μεταφέρονται ακολουθούν ένα συγκεκριμένο πρωτόκολλο.Τα δεδομένα στο δίκτυο συνήθως είναι της μορφής (xml, json). Στην

δικιά μας περίπτωση έχουμε επιλέξει την πιο κοινή μορφή τύπου json. Όπως βλέπουμε στον κώδικα κάνουμε μια μετατροπή των δεδομένων που λαμβάνουμε απο τον sensor σε συμβολοσειρά η οποία όμως είναι τύπου json. Αυτήν την πρακτική την είδαμε και στο http-api του mbed.



2.2.4 Ανάλυση του τμήματος κώδικα που τρέχει στον web server

Ο web server που χρησιμοποιούμε είναι ο apache tomee. Ο συγκεκριμένος web server τρέχει ένα εκτελέσιμο αρχείο .war. Δηλαδή όλη η εφαρμογή μας στο τέλος θα περιέχεται στο τελικό αρχείο .war το οποίο θα φορτώσουμε στον server.

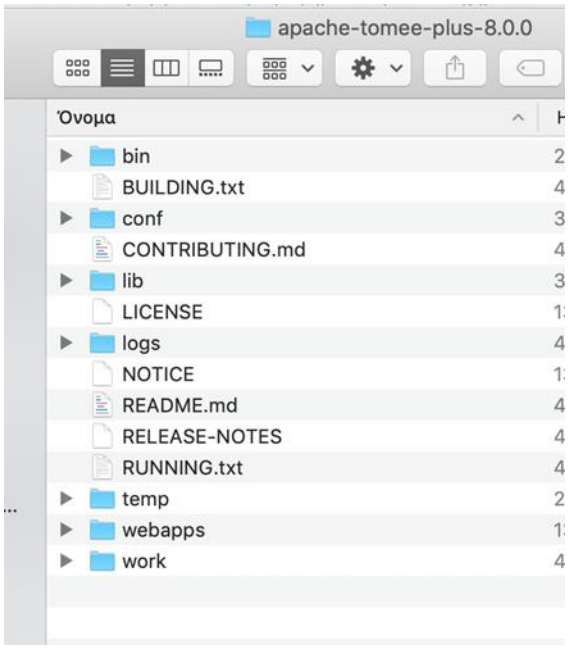


Για την ανάπτυξη της εφαρμογής επιλέξαμε την γλώσσα προγραμματισμού Java. Πιο συγκεκριμένα για την ανάπτυξη τέτοιου είδους web εφαρμογών χρησιμοποιείται ευρέως το framework της Java Enterprise Edition. Για την ευκολότερη διαχείριση της εφαρμογής επιλέξαμε και την χρήση του εργαλείου Maven ώστε να μας απλοποιήσει κάποιες διαδικασίες όπως την αρχικοποίηση της βάσης αλλά και την παραγωγή του εκτελέσιμου αρχείου το οποίο θα φορτωθεί στον web server. Το στήσιμο μιας web εφαρμογής περιλαμβάνει τον συνδυασμό αρκετών τεχνολογιών για αυτόν τον λόγο θα τις χωρίσουμε σε κατηγορίες ανάλογα με την χρησιμότητα που κάνει η κάθεμια.

I. Δομή του web server
II. Εισαγωγή του web server στο περιβάλλον ανάπτυξης Eclipse Ide for Java EE
III. Τμήματα της web εφαρμογής που τρέχουν στον web server
   A. Jpa Hibernate Api
   B. Business logic
   C. Restful Api

2.2.4.1 Δομή του web server

Κατεβάσαμε την portable έκδοση του apache tomee plus.Επιλέξαμε τον συγκεκριμένο server επειδή είναι δωρεάν και χρησιμοποιείται ευρέως για εφαρμογές που είναι γραμμένες σε Java.



Στην εικόνα βλέπουμε τι περιέχει ο συγκεκριμένος web server. Δύο φάκελοι μας ενδιαφέρουν σχετικά με το configuration που θα κάνουμε όσον αφορά την σύνδεση του με την βάση δεδομένων. Ο φάκελος conf περίεχει οτιδηποτε σχετίζεται με το configuration του web server και ο φάκελος lib περιέχει drivers - αρχεία.jar.

Πιο αναλυτικά θα χρειαστεί να τροποποιήσουμε τον αρχείο tomee.xml που βρίσκεται μέσα στον φάκελο conf. Εκεί ουσιαστικά γίνεται η σύνδεση του server με την βάση δεδομένων. Επιπλεόν θα χρειαστεί να κατεβάσουμε απο το ιντερνετ τον driver για την βάση mysql και να τον εισάγουμε μέσα στον φάκελο lib.

tomee.xml

tomee.xml 〉 No Selection

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tomee>
  <!-- see http://tomee.apache.org/containers-and-resources.html -->

  <!-- activate next line to be able to deploy applications in apps -->
  <Deployments dir="apps" />
  <Resource id="java:/phDS" type="DataSource">
    JdbcDriver com.mysql.jdbc.Driver
    JdbcUrl jdbc:mysql://localhost:3306/DbForPhMeter?insecureAuth=true
    JtaManaged true
    UserName userOfPhMeterApplication
    Password 1234
  </Resource>
</tomee>
```

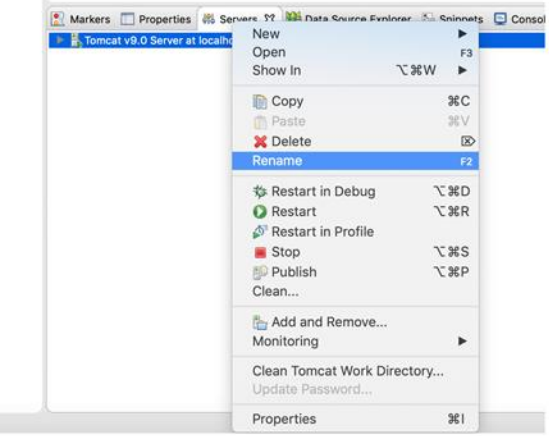
lib

Αναζήτηση

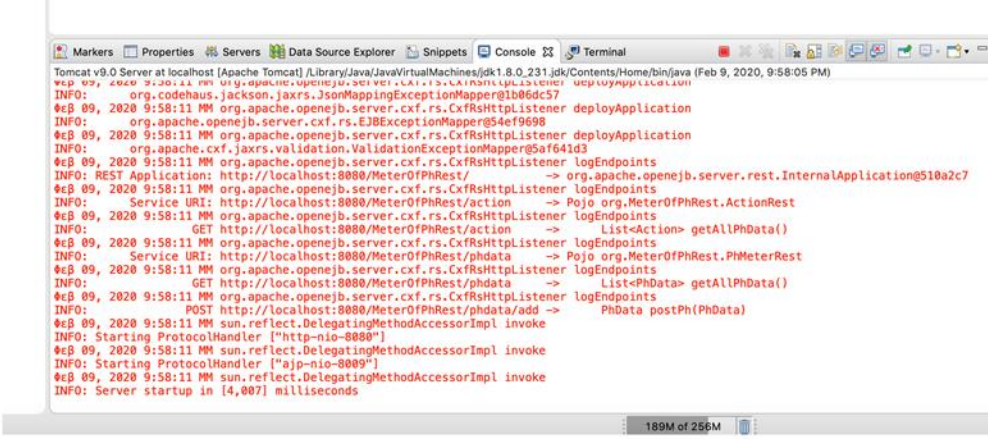
Όνομα	Ημερομηνία τροποποίησης	Μέγεθος	Είδος
microprofile...g-api-1.1.jar	13 Σεπ 2019, 1:31 μμ	10 KB	Αρχείο Java
microprofile...-api-1.2.1.jar	13 Σεπ 2019, 1:31 μμ	21 KB	Αρχείο Java
mimepull-1.9.11.jar	13 Σεπ 2019, 1:31 μμ	67 KB	Αρχείο Java
mp-common-8.0.0.jar	13 Σεπ 2019, 1:31 μμ	21 KB	Αρχείο Java
mp-jwt-8.0.0.jar	13 Σεπ 2019, 1:31 μμ	107 KB	Αρχείο Java
myfaces-api-2.3.4.jar	13 Σεπ 2019, 1:31 μμ	1,3 MB	Αρχείο Java
myfaces-impl-2.3.4.jar	13 Σεπ 2019, 1:31 μμ	2,8 MB	Αρχείο Java
mysql-conn...va-8.0.18.jar	8 Σεπ 2019, 3:09 πμ	2,3 MB	Αρχείο Java
neethi-3.0.3.jar	13 Σεπ 2019, 1:31 μμ	71 KB	Αρχείο Java
openejb-api-8.0.0.jar	13 Σεπ 2019, 1:31 μμ	17 KB	Αρχείο Java
openejb-client-8.0.0.jar	13 Σεπ 2019, 1:31 μμ	353 KB	Αρχείο Java

Αφού τελειώσουμε με τις διαδικασίες που αφορούν το configuration του web server με την βάση είμαστε έτοιμοι να εισάγουμε τον web server στο Eclipse IDE που θα εργαστούμε για να γράψουμε την εφαρμογή. Αυτό το κάνουμε γιατί απο το συγκεριμένο IDE μπορούμε να χειριζόμαστε πιο εύκολα τον web server και έχουμε απευθείας εικόνα των log files του web server στην περίπτωση που πάει κάτι στραβά και "σκάσει" η εφαρμογή.

2.2.4.2 Εισαγωγή του web server στο περιβάλλον ανάπτυξης Eclipse IDE for Java EE



Χειριζόμαστε πολύ εύκολα τις λειτουργίες (έναρξης - παύσης του web server καθώς και την φόρτωση - αφαίρεση του αρχείου.war που περιέχει την εφαρμογή



Πρόσβαση στα log files του web server

2.2.4.3 Τμήματα της εφαρμογής που τρέχουν στον web server

Κάθε ολοκληρωμένη web εφαρμογή αποτελείται απο μία συγκεκριμένη δομή. Σε κάθε τέτοια εφαρμογή συμμετέχουν πάντα οντότητες που είναι υπαρκτές και διαχειριζόμαστε αυτές τις οντότητες. Ομως σε κάθε εφαρμογή τα αποτελέσματα εξάγονται απο τον web server σε άλλες εξωγενείς οντότητες. Συνεπώς διακρίνουμε 3 κατηγορίες όπου κάθεμια έχει την λειτουργικότητα της.

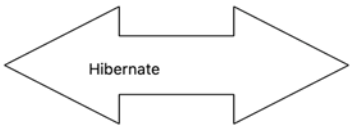
2.2.4.4 JPA Hibernate API

Όπως αναφέραμε και πιο πάνω σε μια εφαρμογή πάντα υπάρχουν οι οντότητες οι οποίες παίζουν καθοριστικό ρόλο. Το εργαλείο που χρησιμοποιούμε για τον λόγο

αυτόν με την βοήθεια του Maven είναι το Hibernate. Ουσιαστικά η επανάσταση του jpa είναι ότι μπορεί να αντιστοιχίσει τα μοντέλα κλάσεις που έχουμε ορίστει στην εφαρμογή με αντίστοιχα tables στην βάση δεδομένων. Το συγκεκριμένο api που χρησιμοποιείται ευρέως μας "γλίτωσε" απο την άσκοπη δημιουργία πολλών queries και πολλά πράγματα όπως είναι φανερό γίνονται αυτόματα. Υπάρχουν πολλών είδη JPA (OpenJPA , Hibernate , EclipseLink). Επιλέξαμε στην εφαρμογή μας το Hibernate.

PhData.java

```java
1  package org.MeterOfPhPersistence;
2
3  import javax.persistence.Column;
11
12 @Entity
13 @NamedQueries({ @NamedQuery(name = "PhData.findAll", query = "SELECT ph FROM PhData ph"),
14         @NamedQuery(name = "PhData.findByDate", query = "SELECT ph FROM PhData ph WHERE ph.date = :
15         @NamedQuery(name = "PhData.findByPhValue", query = "SELECT ph FROM PhData ph WHERE ph.ph =
16 @Table(name = "phdata")
17 public class PhData {
18
19     @Id
20     @GeneratedValue(strategy = GenerationType.IDENTITY)
21     @Column(name = "id")
22     private Long id;
23     @Column(name = "dateOfMeasure")
24     private String date;
25     @Column(name = "timeOfMeasure")
26     private String time;
27     @Column(name = "ph")
28     private double ph;
29
30     public PhData() {
31     }
32
33     public PhData(double ph, String date, String time) {
34         this.ph = ph;
35         this.date = date;
36         this.time = time;
37     }
38
39     public String getDate() {
40         return date;
41     }
42
43     public void setDate(String date) {
44         this.date = date;
45     }
46
47     public String getTime() {
48         return time;
49     }
50
51     public void setTime(String time) {
52         this.time = time;
```




```
Database changed
mysql> show tables;
+------------------------+
| Tables_in_dbforphmeter |
+------------------------+
| actionOfSystem         |
| flyway_schema_history  |
| phdata                 |
+------------------------+
3 rows in set (0,00 sec)

mysql> describe phdata;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| id            | int(11)     | NO   | PRI | NULL    | auto_increment |
| dateOfMeasure | varchar(20) | YES  |     | NULL    |                |
| timeOfMeasure | varchar(20) | YES  |     | NULL    |                |
| ph            | double      | NO   |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
4 rows in set (0,01 sec)

mysql>
```

Mapping class with table in database

Το hibernate μας παρέχει ειδικά annotations που όταν τα δηλώνουμε σε μία κλάση, εκτελούν μια λειτουργία παράλληλα. Για παράδειγμα κάποιες απο αυτές είναι όπως βλέπουμε το "Entity" το οποίο απο την στιγμή που θα το χρησιμοποιήσουμε δηλώνουμε εξαρχής ότι η κλάση μας αποτελεί μια οντότητα. Άλλο ειδικό annotation είναι το "Table" στο οποίο δηλώνουμε με ποιο table της βάσης δεδομένων θα αντιστοιχίσουμε την κλάση μας. Επίσης σε κάθε κλάση η οποία αποτελεί μία οντότητα είναι απαραίτητο να ορίσουμε το αναγνωριστικό που θα έχει κάθε αντικείμενο μας στην βάση και θα το κάνει μοναδικό.Αυτο γίνεται με την προσθήκη του annotation "Id". Όταν θέλουμε να αντιστοίχίσουμε ένα πεδίο της κλάσης με κάποιο column της βάσης δεδομένων γίνεται με την προσθήκη του ειδικού annotation "Column". Ο συνολικός κώδικας βρίσκεται στον φάκελο (MeterOfPhPersistence).

2.2.4.5 Business logic

Είναι το κύριο τμήμα της εφαρμογής.Εκεί ορίζονται οι βασικές λειτουργίες της εφαρμογής.Στην περίπτωση μας επειδή θέλουμε να αποθηκεύουμε μόνο τα δεδομένα στην βάση αλλα και να παίρνουμε αντίστοιχα απο την βάση δεδομένα ,περιέχει μόνο queries προς την βάση δεδομένων.Ο συνολικός κώδικας βρίσκεται στον φάκελο (MeterOfPhService).

2.2.4.6 Restful API

Επειδή ο web server όπως φαίνεται στο αρχικό διάγραμμα ροής θέλουμε να επικοινωνεί με τον microcontroller αλλα και το front-end κομμάτι είναι ανάγκη να δημιουργήσουμε ένα interface το οποίο παρέχει συνδέσμους που μπορούν αντίστοιχα εξωγενείς οντότητες να πάρουν δεδομένα και να στείλουν δεδομένα στα συγκεκριμένα άκρα ,τα λεγόμενα endpoints. Αυτό έγινε με την βοήθεια των jax-ws web services. Πάλι το συγκεκριμένο εργαλείο εισάγεται με την βοήθεια του Maven. Τα συγκεκριμένα services μας προσφέρουν και αυτά ειδικα annotations τα οποία εκτελούν και αυτά κάποιες ειδικές λειτουργίες.

```
1  package org.MeterOfPhRest;
2
3⊕ import java.text.SimpleDateFormat;
9
0  @Path("/phdata")
1  public class PhMeterRest {
2
3⊖     @Inject
4      PhMeterService phMeterService;
5
5⊖     @GET
7      @Produces({ MediaType.APPLICATION_JSON })
8      public List<PhData> getAllPhData() {
9          return phMeterService.findAll();
0      }
1
2⊖     @Path("/add")
3      @POST
4      @Consumes({MediaType.APPLICATION_JSON})
5      public PhData postPh(PhData phData) {
5          phData.setDate(new SimpleDateFormat("yyyy-MM-dd").format(Calendar.getInstance().getTime()));
7          phData.setTime(new SimpleDateFormat("HH:mm:ss").format(Calendar.getInstance().getTime()));
8          return phMeterService.save(phData);
9      }
0  }
1
```

```
INFO:      org.apache.openejb.server.cxf.rs.EJBExceptionMapper@54ef9698
Φεβ 09, 2020 9:58:11 MM org.apache.openejb.server.cxf.rs.CxfRsHttpListener deployApplication
INFO:      org.apache.cxf.jaxrs.validation.ValidationExceptionMapper@5af641d3
Φεβ 09, 2020 9:58:11 MM org.apache.openejb.server.cxf.rs.CxfRsHttpListener logEndpoints
INFO: REST Application: http://localhost:8080/MeterOfPhRest/          -> org.apache.openejb.server.rest.InternalApplica
Φεβ 09, 2020 9:58:11 MM org.apache.openejb.server.cxf.rs.CxfRsHttpListener logEndpoints
INFO:      Service URI: http://localhost:8080/MeterOfPhRest/action     -> Pojo org.MeterOfPhRest.ActionRest
Φεβ 09, 2020 9:58:11 MM org.apache.openejb.server.cxf.rs.CxfRsHttpListener logEndpoints
INFO:              GET http://localhost:8080/MeterOfPhRest/action     ->      List<Action> getAllPhData()
Φεβ 09, 2020 9:58:11 MM org.apache.openejb.server.cxf.rs.CxfRsHttpListener logEndpoints
INFO:      Service URI: http://localhost:8080/MeterOfPhRest/phdata     -> Pojo org.MeterOfPhRest.PhMeterRest
Φεβ 09, 2020 9:58:11 MM org.apache.openejb.server.cxf.rs.CxfRsHttpListener logEndpoints
INFO:              GET http://localhost:8080/MeterOfPhRest/phdata     ->      List<PhData> getAllPhData()
Φεβ 09, 2020 9:58:11 MM org.apache.openejb.server.cxf.rs.CxfRsHttpListener logEndpoints
INFO:             POST http://localhost:8080/MeterOfPhRest/phdata/add ->      PhData postPh(PhData)
Φεβ 09, 2020 9:58:11 MM sun.reflect.DelegatingMethodAccessorImpl invoke
```

Για παράδειγμα για να ορίσουμε ένα συγκεκριμένο “άκρο”, αυτό θα γίνει με την βοήθεια του annotation “Path”. Αντίστοιχα για να ορίσουμε τον τύπο της μεθόδου (post,get) ορίζουμε και τα αντίστοιχα annotations. Ο συνολικός κώδικας βρίσκεται στον φάκελο (MeterOfPhRest) .

2.2.5 Ανάλυση του τμήματος κώδικα που εκτελείται στην βάση δεδομένων

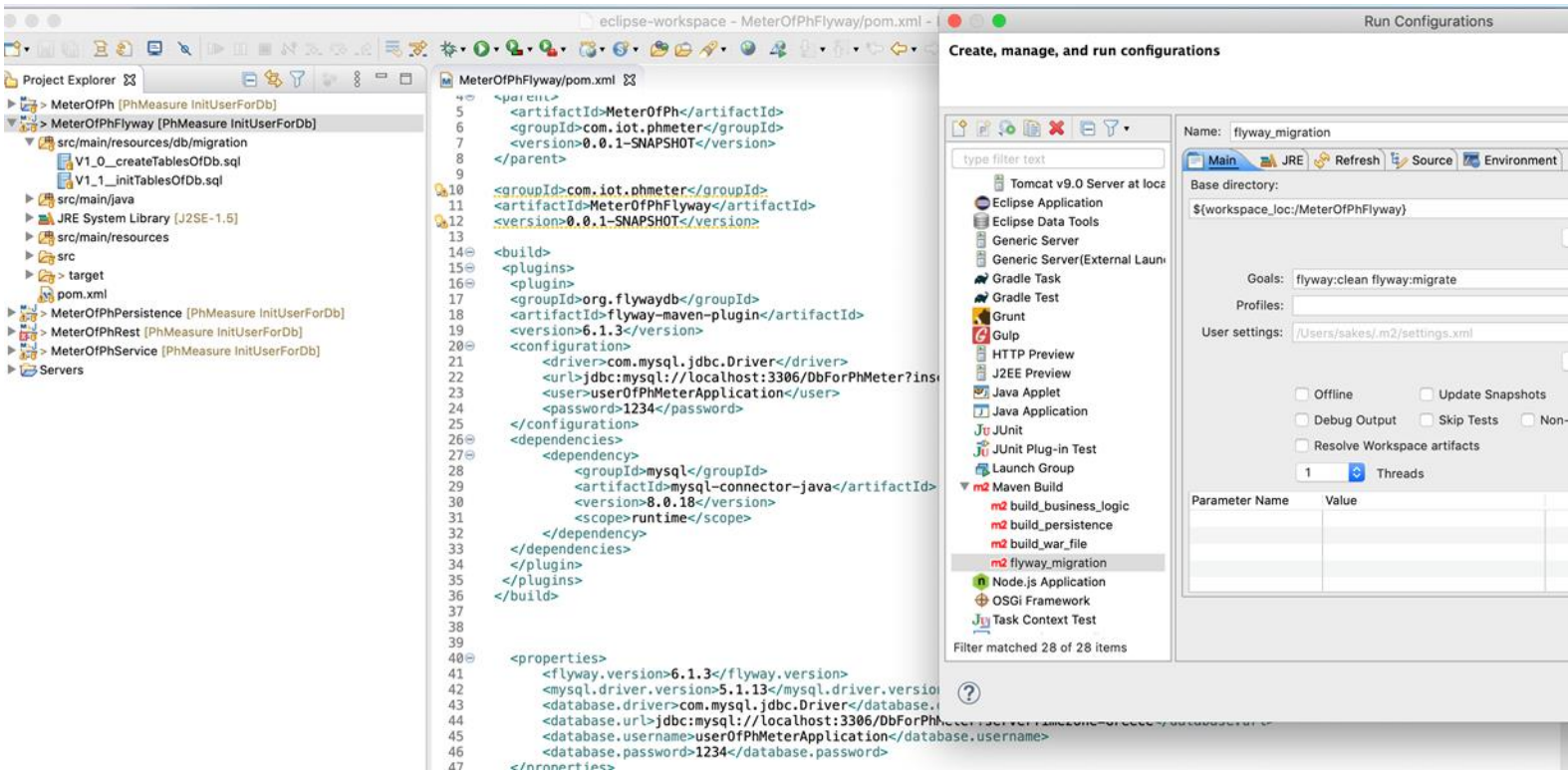
Για την συγκεκριμένη εφαρμογή χρησιμοποιήσαμε την mysql version 8.0.18. Αρχικά για την δημιουργία της βάσης και του χρήστη που θα έχει πρόσβαση σε αυτήν, τρέχουμε το script που φαίνεται στην εικόνα παρακάτω. Πάντα σε κάθε δεδομένων πρέπει να δημιουργούμε πρώτα την βάση μας δίνοντας απλά ένα όνομα αλλά και τον χρήστη που θα έχει πρόσβαση σε αυτήν.

```
Users > sakes > Documents > GitHub > initUserForDB > createUserForDbOfPhMeter.sql
1  -- run in mysql console: source /Users/sakes/Documents/GitHub/PhMeasure/initUserForDB/createUserForDbOfPhMeter.sql
2  DROP USER IF EXISTS 'userOfPhMeterApplication'@'localhost';
3  DROP DATABASE IF EXISTS DbForPhMeter;
4  CREATE DATABASE DbForPhMeter;
5  CREATE USER 'userOfPhMeterApplication'@'localhost' IDENTIFIED WITH mysql_native_password  BY'1234';
6  GRANT ALL PRIVILEGES ON DbForPhMeter.* TO 'userOfPhMeterApplication'@'localhost' IDENTIFIED WITH mysql_native_password  BY'1234';
7  FLUSH PRIVILEGES;
```

Όπως βλέπουμε το συγκεκριμένο αρχείο είναι τύπου .sql. Οι βάσεις δεδομένων αναγνωρίζουν αυτού του είδους script-files .sql. Οπότε τώρα αφού έχουμε δημιουργήσει την βάση είμαστε έτοιμοι να προχωρήσουμε στην δημιουργία των tables που θέλουμε για την εφαρμογή μας και στην συνέχεια να γεμίσουμε τα tables αυτά με δεδομένα. Μπορούμε να κάνουμε την διαδικασία δηλ να δημιουργούμε τα tables “χειροκίνητα” αντί όμως για αυτό θα προτιμήσουμε μία άλλη πιο απλή διαδικασία. Το maven πάλι μας προσφέρει ένα ειδικό εργαλείο που λέγεται “flyway”. Το flyway μας δίνει την δυνατότητα με την εκτέλεση κάποιων εντολών να δημιουργεί αυτό τα tables στην βάση να τα γεμίζει δεδομένα αλλά και να τα διαγράφει.

- Flyway clean -> Διαγράφει όλα τα tables της βάσης
- Flyway migrate -> Δημιουργεί όλα τα τα tables στην βάση και αναλαμβάνει και την αρχικοποίηση τους.

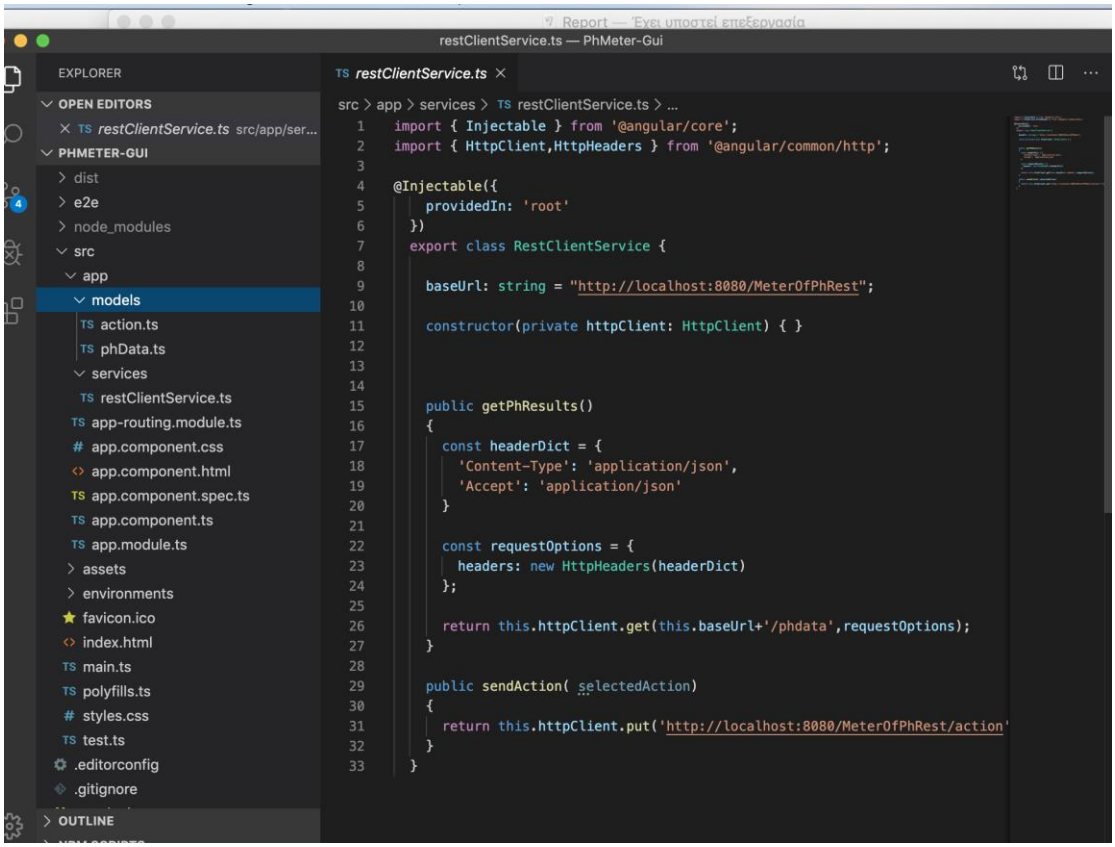
Υπάρχει απλά μία σύμβαση στα αρχεία .sql που χρησιμιοποιεί το flyway. Θα πρέπει στο όνομα τους να βάλουμε το “version” ώστε να ξέρει το flyway ποιο αρχείο θα τρέξει πρώτο.

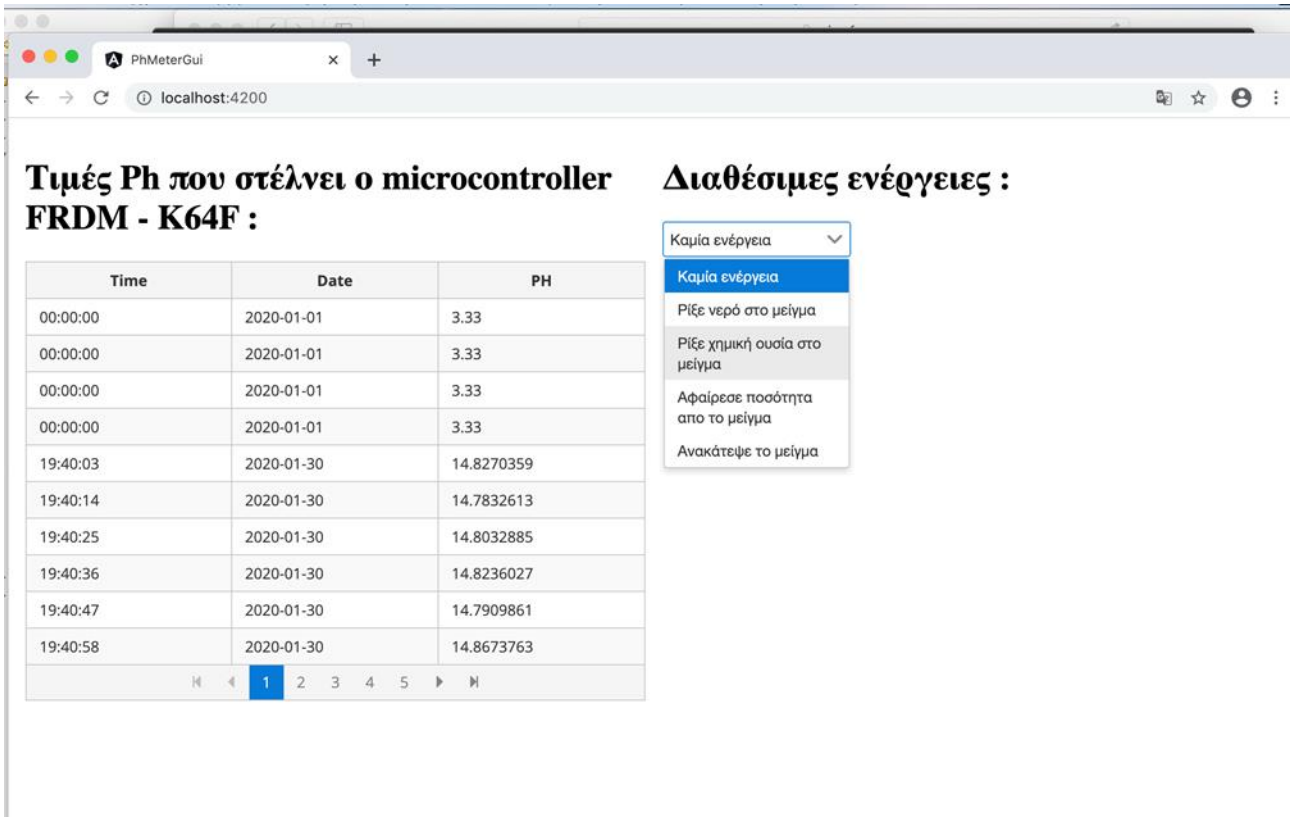


Ο κώδικας που περιέχει το αρχεία createTablesOfDb σχετίζεται με την δημιουργία των tables αλλα και των column που περιέχει το κάθε table. Αυτό που μας ενδιαφέρει σε κάθε column είναι να ορίσουμε τον τύπο δεδομένων που θα περιέχει το καθένα ,το όνομα του column αλλά και την μεταξύ τους σχέση που μπορεί να έχει ένα column ενος table με άλλο column άλλου table (ξένα κλειδιά). Ο κώδικας που περιέχει το αρχείο initTablesOfDb σχετίζεται με το "γέμισμα " δεδομένων στην βάση. Προσθέτουμε σε κάθε table εγγραφές απλά πρέπει να προσέξουμε οι τύποι των δεδομένων που ορίσαμε στο προηγούμενο αρχείο να συμβαδίζουν. Ο συνολικός κώδικας βρίσκεται στον φάκελο (MeterOfPhFlyway).

2.2.6 Ανάλυση του τμήματος κώδικα που εκτελείται στο front-end

Χρησιμοποιήσαμε για το front-end το τελικό τμήμα που θα βλέπει ο χρήστης την τεχνολογία angular javascript διότι προσφέρεται δωρεάν αυτή, αλλά και το περιβάλλον ανάπτυξης που χρησιμοποιήσαμε, το Visual Studio Code. Ο συνολικός κώδικας βρίσκεται μέσα στον φάκελο (PhMeterGui). Τα περισσότερα αρχεία στον φάκελο αυτόν δημιουργήθηκαν αυτόματα όταν δημιουργήσαμε το angular project. Απο εκεί και πέρα τα δύο αρχεία που μας ενδιαφέρουν είναι τα app.component.html και app.component.ts. Εκεί δημιουργούμε ουσιαστικά το datatable το οποίο είναι δυναμικό και παίρνει τα δεδομένα που έρχονται απο τον web server.





Time	Date	PH
00:00:00	2020-01-01	3.33
00:00:00	2020-01-01	3.33
00:00:00	2020-01-01	3.33
00:00:00	2020-01-01	3.33
19:40:03	2020-01-30	14.8270359
19:40:14	2020-01-30	14.7832613
19:40:25	2020-01-30	14.8032885
19:40:36	2020-01-30	14.8236027
19:40:47	2020-01-30	14.7909861
19:40:58	2020-01-30	14.8673763

Μέρος Τρίτο

3.1 Εισαγωγή

Για το τρίτο μέρος της εργασίας μας πρόκειται να προτείνουμε βάση δικών μας υπολογισμών τις κατάλληλες τιμές τόσο για το resolution range όσο και για το datarate range αλλά και το power consumption του ph αισθητήρα που χρησιμοποιήσαμε στην υλοποίηση του iot project μας, έτσι ώστε να εμπλουτίσουμε τα δεδομένα της ήδη υπάρχουσας android εφαρμογής μας.

3.2 Υπολογισμός του resolution range.

Για τον υπολογισμό του resolution range του αισθητήρα ph παρατηρήσαμε ότι ο αισθητήρας στέλνει αναλογικό σήμα (μέσω της θύρας A0 του board) και άρα το resolution range του αισθητήρα εξαρτάται αποκλειστικά από τον Analog to Digital Converter του board. Από το datasheet του microcontroller FRDM-k64f που χρησιμοποιούμε για την υλοποίηση του project, βλέπουμε ότι η A0 θύρα έχει 16 bits resolution range. Οπότε το resolution range του αισθητήρα μας είναι και αυτό 16 bits.

3.3 Υπολογισμός του datarate range.

Για τον υπολογισμό του datarange range του αισθητήρα μας, παρατηρήσαμε ότι στο datasheet του χρειάζεται ένα με δύο λεπτά για να κάνει την κάθε μέτρηση. Επομένως το datarate range του αισθητήρα θα είναι ίσο με: $datarate\ range = \frac{resolution\ range}{time}$. Οπότε στην περίπτωσή μας, εφόσον η μέτρηση διαρκεί από ένα έως δύο λεπτά, θα υπολογίσουμε το datarate_min παίρνοντας ως $time = 120sec$ και το datarate_max παίρνοντας ως $time = 60\ sec$. Συνεπώς έχουμε: $datarate_\min = \frac{16}{120} = 0{,}133\ bps$ και αντίστοιχα $datarate_\max = \frac{16}{60} = 0{,}266\ bps$.

3.4 Υπολογισμός του ADC(Analog to Digital Conversion) power.

Για τον υπολογισμό της ισχύος που καταναλώνεται για την μετατροπή του Αναλογικού σήματος σε Ψηφιακό χρησιμοποιήσαμε σαν δεδομένο την υπόθεση από το πρώτο κεφάλαιο του βιβλίου “Enabling the internet of things” του “Massimo Alioto”. Στην υπόθεση αυτή ο συγγραφέας ισχυρίζεται ότι απαιτούνται 30fJ για κάθε βήμα κατά την μετατροπή του σήματος. Άρα μ’ αυτόν τον τρόπο η ισχύς για την μετατροπή μπορεί να υπολογιστεί από τον τύπο: $ADC = power\ for\ step * datarate$. Οπότε στην περίπτωσή μας έχουμε: $ADC_\min = 30 * 0{,}133 = 4\ fJ$ και αντίστοιχα $ADC_\max = 30 * 0.266 = 8\ fJ$.

3.5 Εισαγωγή των δεδομένων στην Android εφαρμογή μας.

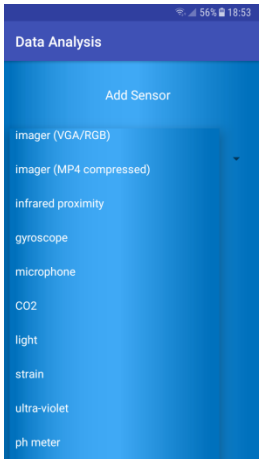
Έχοντας πλέον τις μετρήσεις για τον αισθητήρα που χρησιμοποιήσαμε, μπορούμε τώρα να εισάγουμε στην κλάση “Calculate” ένα ακόμη αντικείμενο “Sensors” για τον αισθητήρα ph, όπως βλέπουμε και στην παρακάτω εικόνα.

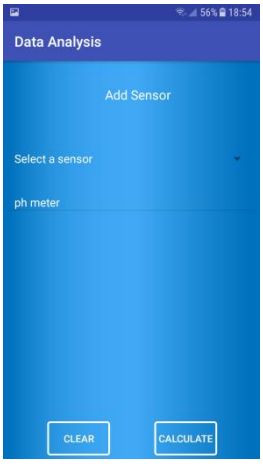
```
public class Calculate extends AppCompatActivity {

    ArrayList<Sensors> sensors = new ArrayList();
    int resolution_min=0, resolution_max=0;
    double datarate_min=0, datarate_max=0;
    double adc_min=0, adc_max=0;
    double total_min=0, total_max=0;



    @Override
    protected void onCreate (Bundle savedInstanceState) {

        super.onCreate( savedInstanceState );
        setContentView( R.layout.calculate );

        Intent intent = getIntent();

        final ArrayList<String> list = getIntent().getStringArrayListExtra( name: "items");


        sensors.add( new Sensors( name: "heart rate",  resolution_min: 4,  resolution_max: 12,  datarate_min: 5000,  datarate_max: 10000,  adc_min: 0.000000001,  adc_max: 0.000000005 ) );
        sensors.add( new Sensors( name: "humidity",  resolution_min: 8,  resolution_max: 16,  datarate_min: 10,  datarate_max: 50,  adc_min: 0,  adc_max: 0 ) );
        sensors.add( new Sensors( name: "battery monitor",  resolution_min: 5,  resolution_max: 8,  datarate_min: 5,  datarate_max: 30,  adc_min: 0,  adc_max: 0 ) );
        sensors.add( new Sensors( name: "temperature",  resolution_min: 6,  resolution_max: 12,  datarate_min: 100,  datarate_max: 300,  adc_min: 0.0000000005,  adc_max: 0.000000001 ) );
        sensors.add( new Sensors( name: "accelerometer",  resolution_min: 8,  resolution_max: 14,  datarate_min: 2000,  datarate_max: 5000,  adc_min: 0.00000001,  adc_max: 0.00000003 ) );
        sensors.add( new Sensors( name: "magnetometer",  resolution_min: 8,  resolution_max: 16,  datarate_min: 800,  datarate_max: 1200,  adc_min: 0.000000007,  adc_max: 0.00000001 ) );
        sensors.add( new Sensors( name: "altimeter/pressure",  resolution_min: 8,  resolution_max: 24,  datarate_min: 800,  datarate_max: 2000,  adc_min: 0.000000007,  adc_max: 0.00000003 ) );
        sensors.add( new Sensors( name: "imager (VGA/RGB)",  resolution_min: 8,  resolution_max: 10,  datarate_min: 100000000,  datarate_max: 200000000,  adc_min: 0.0007,  adc_max: 0.001 ) );
        sensors.add( new Sensors( name: "imager (MP4 compressed)",  resolution_min: 8,  resolution_max: 10,  datarate_min: 10000000,  datarate_max: 300000000,  adc_min: 0.0007,  adc_max: 0.001 ) );
        sensors.add( new Sensors( name: "infrared proximity",  resolution_min: 10,  resolution_max: 16,  datarate_min: 300,  datarate_max: 800,  adc_min: 0.000000007,  adc_max: 0.000000013 ) );
        sensors.add( new Sensors( name: "gyroscope",  resolution_min: 12,  resolution_max: 16,  datarate_min: 10000,  datarate_max: 3000,  adc_min: 0.000002,  adc_max: 0.000004 ) );
        sensors.add( new Sensors( name: "microphone",  resolution_min: 12,  resolution_max: 16,  datarate_min: 200000,  datarate_max: 50000,  adc_min: 0.00002,  adc_max: 0.00008 ) );
        sensors.add( new Sensors( name: "CO2",  resolution_min: 14,  resolution_max: 16,  datarate_min: 300,  datarate_max: 700,  adc_min: 0.0000000004,  adc_max: 0.0000000006 ) );
        sensors.add( new Sensors( name: "light",  resolution_min: 16,  resolution_max: 24,  datarate_min: 200,  datarate_max: 500,  adc_min: 0.0000005,  adc_max: 0.0000008 ) );
        sensors.add( new Sensors( name: "strain",  resolution_min: 16,  resolution_max: 24,  datarate_min: 20000,  datarate_max: 50000,  adc_min: 0.00005,  adc_max: 0.00008 ) );
        sensors.add( new Sensors( name: "ultra-violet",  resolution_min: 0,  resolution_max: 0,  datarate_min: 400,  datarate_max: 700,  adc_min: 0.0000001,  adc_max: 0.0000002 ) );


        //Add the Ph Sensor
        sensors.add( new Sensors( name: "ph meter",  resolution_min: 16,  resolution_max: 16,  datarate_min: 0.133,  datarate_max: 0.266,  adc_min: 0.00000000000004,  adc_max: 0.000000000000008 ));
```

Επειδή μέχρι τώρα είχαμε ακέραιες τιμές στα datarate_min και datarate_max και είχαμε χρησιμοποιήσει τον τύπο long int, τώρα καθώς οι τιμές που υπολογίσαμε για τον αισθητήρα του ph είναι δεκαδικές, χρειάστηκε να αλλάξουμε τον τύπο των μεταβλητών σε double. Ακόμη, θα χρειαστεί να προσθέσουμε την επιλογή για τον αισθητήρα ph στην λίστα με τους διαθέσιμους αισθητήρες που βλέπει ο χρήστης στην εφαρμογή.

```
package sotiris.zogos.dataanalysis;

import ...

public class MainActivity extends AppCompatActivity {

    ArrayList<String> list = new ArrayList<String>();
    ArrayAdapter<String> adapter,listadapter;

    @Override
    protected void onCreate (Bundle savedInstanceState) {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_main );

        final Spinner dropdown = findViewById(R.id.spinner1);
        String[] items = new String[] {"Select a sensor", "heart rate", "humidity", "battery monitor", "temperature", "accelerometer", "magnetometer", "altimeter/pressure",
                "imager (VGA/RGB)", "imager (MP4 compressed)", "infrared proximity", "gyroscope", "microphone", "CO2", "light", "strain", "ultra-violet", "ph meter"};


        ArrayAdapter<String> adapter = new ArrayAdapter<>( context: this, android.R.layout.simple_spinner_dropdown_item, items);
        dropdown.setAdapter(adapter);


        ListView lstView = (ListView) findViewById(R.id.listView1);
        listadapter = new ArrayAdapter<String> ( context: this, android.R.layout.simple_spinner_item,list);
                lstView.setAdapter(listadapter);
                dropdown.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1,
                                       int arg2, long arg3) {
                // TODO Auto-generated method stub
                String s = dropdown.getItemAtPosition(arg2).toString();
                if (!(s.matches( regex: "Select a sensor" ))) {
                    list.add( s );
                    dropdown.setSelection( 0 );
                    listadapter.notifyDataSetChanged();
                }
            }

            @Override
            public void onNothingSelected(AdapterView<?> arg0) {
                // TODO Auto-generated method stub
            }
```

Τώρα η εφαρμογή μας είναι ενημερωμένη, ο αισθητήρας ph βρίσκεται στην λίστα με τους διαθέσιμους προς επιλογή, και τα χαρακτηριστικά του έχουν περαστεί στο σύστημα για οποιονδήποτε υπολογισμό. Παρακάτω παραθέτουμε ένα παράδειγμα εκτέλεσης της ενημερωμένης μας εφαρμογής.





These are the selected sensors:
ph meter
This is your analysis results
Minimum resolution: 16 bits
Maximum resolution: 16 bits
Minimum datarate: 0.133 bps
Maximum datarate: 0.266 bps
Minimum ADC power: 4.0E-15 W
Maximum ADC power: 8.0E-15 W
Minimum total power: 6.650040000000001E-10 W
Maximum total power: 1.3300080000000002E-9 W