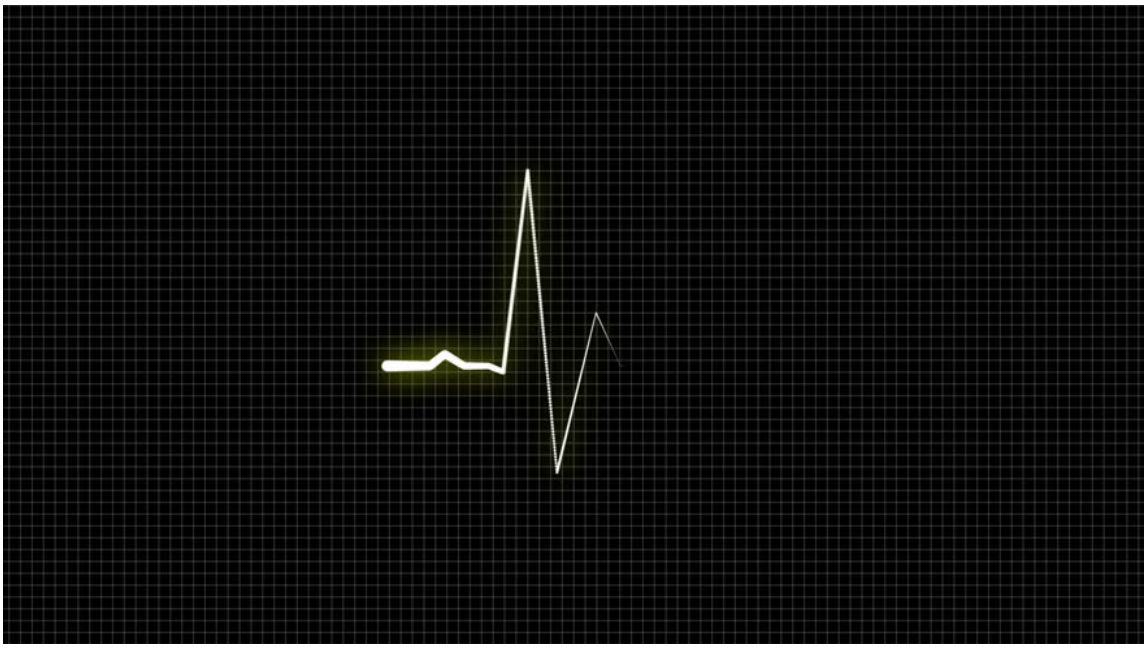
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2016-2017

Βιοϊατρική Τεχνολογία

7ο εξάμηνο



Θέμα Β:

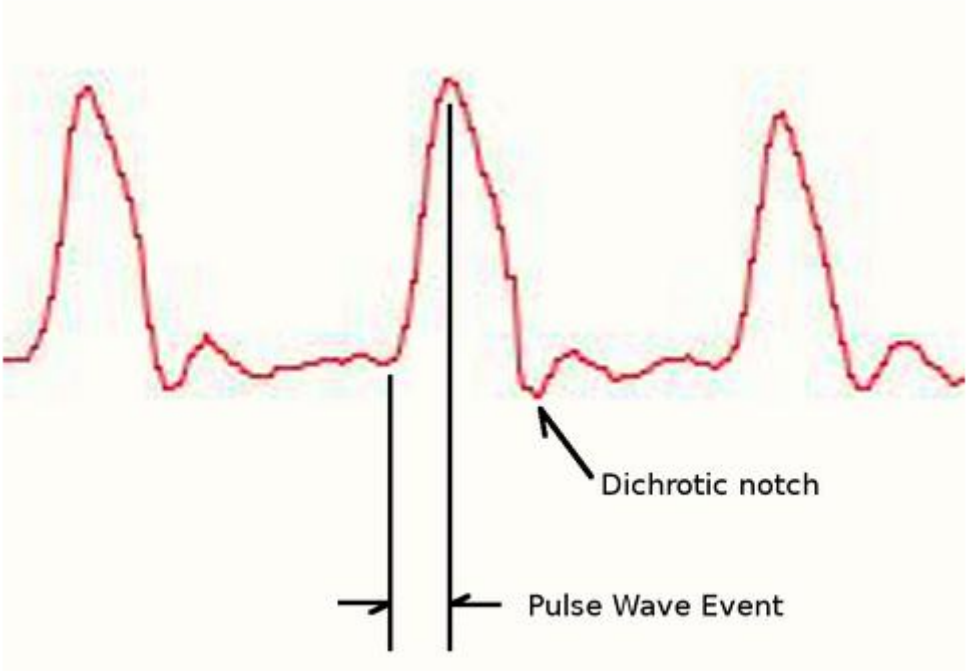
Κατασκευή συστήματος ανίχνευσης/ελέγχου του καρδιακού ρυθμού και δημιουργία μιας εφαρμογής για smartphone για την απεικόνιση των σημάτων, ιστορικού και συναγερμού.

Τσουβάλας Βασίλης 8040

Περιεχόμενα

1. Κατασκευή συστήματος ανίχνευσης καρδιακού παλμού

Για την κατασκευή του συστήματος αρχικά χρησιμοποιούμε έναν φωτοπληθυσμογράφο (PPG), μια μη επεμβατική τεχνική μέσω της οποίας σε επιλεγμένα σημεία του σώματος εντοπίζονται αλλαγές στην ροή του αίματος κατά τον καρδιακό κύκλο. Στην προκειμένη περίπτωση χρησιμοποιείται σε συνδυασμό με μία διαπεραστική φωτεινή ακτινοβολία η οποία παράγεται συνήθως από LED και ανιχνεύεται από έναν φωτοανιχνευτή τοποθετημένο απέναντι από την πηγή φωτός ή ακριβώς δίπλα από την πηγή. Ουσιαστικά αυτή η συσκευή φωτίζει το δέρμα και μετρά τις αλλαγές στην απορρόφηση του φωτός που προκύπτουν ανάλογα με τον όγκο του αίματος στο δέρμα. Για την λήψη του σήματος του καρδιακού παλμού σε πραγματικό χρόνο από τον υπολογιστή χρησιμοποιούνται οξύμετρα που τοποθετούνται στο δάχτυλο και συλλέγουν δεδομένα μέσω του PPG. Το σήμα που λαμβάνουμε είναι μια αναλογική διακύμανση σε volt και έχει μια τυπική μορφή που δείχνουμε παρακάτω. Ο ανιχνευτής παλμού που χρησιμοποιούμε ενισχύει το σήμα και κανονικοποιεί την κυματομορφή γύρω από το μέσο της τροφοδοσίας. Αν η ποσότητα του φωτός που ανακλάται παραμένει σταθερή το σήμα παίρνει τιμές γύρω στο 512, με περισσότερο φως η τιμή ανεβαίνει και το αντίθετο.



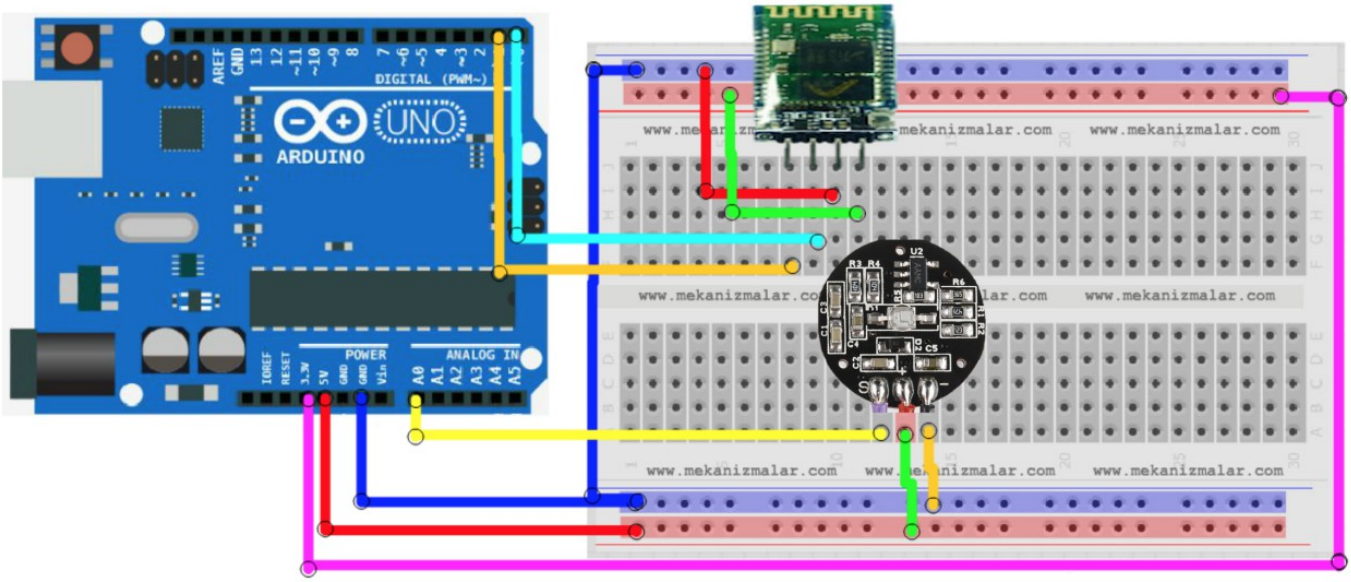
Τυπική κυματομορφή του PPG

Στόχος μας είναι να βρούμε τις στιγμές που συμβαίνει ο παλμός πετυχαίνοντας το ίδιο σημείο της κυματομορφής και έπειτα να μετρήσουμε το διάστημα που πέρασε. Αυτή η περίοδος ονομάζεται IBI και χρησιμοποιείται στον υπολογισμό του BPM (καρδιακοί παλμοί το λεπτό).

Υπάρχουν διάφορες απόψεις σχετικά με το πότε συμβαίνει ο καρδιακός παλμός . Στην παρούσα εργασία θεωρούμε ότι συμβαίνει στο μέσο του πλάτους amp , που ορίζεται ως η απόσταση της χαμηλότερης τιμής του διαγράμματος (εξαιρώντας το dichroic notch, μια προς τα κάτω εκτροπή που παρουσιάζεται στο διάγραμμα και συμβαίνει ακριβώς μετά την κορυφή της συστολής) και της υψηλότερης τιμής του διαγράμματος.

2. Συνδεσμολογία

Για μια αξιόπιστη μέτρηση χρησιμοποιούμε το Arduino Uno ATmega328P με 16MHz clock, το οποίο ρυθμίζουμε ώστε να έχει baut rate 115200 και να κάνει δειγματοληψία ανά 4 msec. Παρακάτω φαίνεται μια εικόνα της συνολικής συνδεσμολογίας:



Χρησιμοποιήθηκαν για την κατασκευή του συστήματος:

- ο αισθητήρας των καρδιακών παλμών,
- το Arduino Uno,
- ένα Bluetooth Module (HC-06 RS232),
- πλακέτα και καλώδια, μπαταρία 9V για φορητή χρήση του Arduino,
- ένα Android smartphone.

Συνδέουμε στο Arduino Uno τον ανιχνευτή μας, ο οποίος συνδέεται με την τροφοδοσία 5V που παρέχει το Arduino, την γείωση και την αναλογική είσοδο A0. Το Bluetooth module συνδέεται επίσης με την τροφοδοσία 3.3V, και το Rx δέκτη στο Tx του Arduino, το Tx στο Rx του Arduino.

Η καλωδίωση χρησιμοποιείται για διευκόλυνση, αρχικά για την επιμήκυνση του ανιχνευτή μας, ώστε να μην έχουμε μεγάλο περιορισμό στην απόσταση που πρέπει να βρισκόμαστε από τον μικροεπεξεργαστή, και κατά δεύτερον, για την εύκολη αποσύνδεση του Bluetooth Module (απαραίτητη διαδικασία για την εκ νέου φόρτωση του κώδικα στο Arduino).

Σύμφωνα με τον κατασκευαστή του αισθητήρα, είναι απαραίτητο να μεριμνήσουμε για την προστασία του αισθητήρα από τον ιδρώτα και έλαια που μπορεί να υπάρχουν στα δάχτυλα ή στο σημείο μέτρησης του ανθρώπινου σώματος, διότι επηρεάζουν αρνητικά την ποιότητα του σήματος. Ακόμα, είναι σημαντικό να προστατευτεί και το ανθρώπινο σώμα από τον ηλεκτρισμό που διαπερνά τον αισθητήρα. Για το λόγο αυτό περιτυλίξαμε τον αισθητήρα με πλαστικό διαφανές υλικό (σιλοτέιπ).

3. Ανάλυση του κώδικα που φορτώθηκε στο Arduino

3.1 Αρχικοποίηση συστήματος - Εκτύπωση συμβόλων και τιμών

Αρχικά επιλέγουμε την αναλογική θύρα εισόδου A0 του Arduino, καθώς και τις μεταβλητές εισόδου και εξόδου του Arduino. Επίσης δίνεται μια ενδεικτική αρχική τιμή του IBI. Η μεταβλητή Signal θα κρατάει τα εισερχόμενα δεδομένα. Η μεταβλητή BPM θα κρατάει τον υπολογισμό του καρδιακού παλμού. Η μεταβλητή QS κρατάει την κατάσταση true όταν το Arduino ανιχνεύει έναν παλμό. Η μεταβλητή Pulse γίνεται true όταν διακρίνουμε καρδιακό παλμό.

```
//  Variables
int pulsePin = 0;                 // Pulse Sensor purple wire connected to analog pin 0

// Volatile Variables, used in the interrupt service routine!
volatile int BPM;                   // int that holds raw Analog in 0. updated every 4ms
volatile int Signal;                // holds the incoming raw data
volatile int IBI = 800;             // int that holds the time interval between beats! Must be seeded!
volatile boolean Pulse = false;     // "True" when User's live heartbeat is detected. "False" when not a "live beat".
volatile boolean QS = false;        // becomes true when Arduoino finds a beat.
```

Ορίζουμε το baud rate δηλαδή τον ρυθμό μετάδοσης του Arduino, και ο οποίος πρέπει να συμφωνεί με το baud rate του Bluetooth module. Αυτή είναι μια ξεχωριστή διαδικασία που γίνεται πάλι μέσω του Arduino. Τέλος καλούμε την ρουτίνα διακοπής που θα αναλυθεί στη συνέχεια.

```
void setup(){
  Serial.begin(115200);             // we agree to talk fast!
  interruptSetup();                 // sets up to read Pulse Sensor signal every 4ms
}
```

Με τον παρακάτω κώδικα στέλνουμε τα δεδομένα εξόδου που θα διαβαστούν στην προκειμένη περίπτωση από το Bluetooth Module που είναι συνδεμένο στο Tx του Arduino. Καλείται επανειλημμένα η ρουτίνα serialOutput() με την οποία στέλνονται οι τιμές του δυναμικού διακύμανσης με πρόθεμα ένα σύμβολο S. Όταν βρεθεί παλμός το QS παίρνει την τιμή true και καλείται η ρουτίνα εξόδου serialOutputWhenBeatHappens() με την οποία τυπώνεται το σύμβολο B και στην συνέχεια η τιμή του BPM που υπολογίστηκε και το συμβόλο Q με την τιμή IBI που υπολογίστηκε. Η αποστολή δεδομένων γίνεται με μια καθυστέρηση 25 ms ώστε να μην δημιουργηθούν προβλήματα. Προσθέτουμε και την ρουτίνα sendDataToSerial() που πραγματοποιεί την εμφάνιση των συμβόλων και αριθμών στην κονσόλα, για σκοπούς debugging.

```
//  Where the Magic Happens
void loop(){

    serialOutput() ;

  if (QS == true){     // A Heartbeat Was Found
                       // BPM and IBI have been Determined
                       // Quantified Self "QS" true when arduino finds a heartbeat

        serialOutputWhenBeatHappens();   // A Beat Happened, Output that to serial.
        QS = false;                      // reset the Quantified Self flag for next time
  }

  delay(25);                             //  take a break
}
```

```
void serialOutput(){   // Output Serial.

      sendDataToSerial('S', Signal);     // goes to sendDataToSerial function

}


//  Decides How To OutPut BPM and IBI Data
void serialOutputWhenBeatHappens(){

        sendDataToSerial('B',BPM);   // send heart rate with a 'B' prefix
        sendDataToSerial('Q',IBI);   // send time between beats with a 'Q' prefix

}


//  Sends Data to Pulse Sensor Processing App, Native Mac App, or Third-party Serial Readers.
void sendDataToSerial(char symbol, int data ){
    Serial.print(symbol);
    Serial.print(",");
    Serial.print(data);
    Serial.print(",");

  }
```

3.2 Σύστημα ανίχνευσης καρδιακού παλμού - Ρουτίνα διακοπής

Αρχικά γίνεται η αρχικοποίηση των μεταβλητών που θα χρησιμοποιηθούν:

- Ο πίνακας rate θα κρατάει τις 10 πιο πρόσφατες τιμές του IBI, για μεγαλύτερη αξιοπιστία στην μέτρηση του BPM,
- Η μεταβλητή sampleCounter κρατάει την τρέχουσα στιγμή δειγματοληψίας, αρχικοποιείται στο 0 και κάθε φορά αυξάνεται κατά 4 εφόσον η ρουτίνα διακοπής καλείται κάθε 4ms,
- Το lastBeatTime χρησιμοποιείται στον υπολογισμό IBI = sampleCounter – lastBeatTime μόλις ανιχνευτεί παλμός, αφού πρόκειται για το διάστημα μεταξύ δύο διαδοχικών παλμών,
- Έχουμε τις τιμές P, T, thresh και amp που αναφέρονται σε μεγέθη του παλμού, την κορυφή, το ελάχιστο σημείο, το μέσο της κυματομορφής και το πλάτος αντίστοιχα, τα οποία αρχικοποιούνται σε ενδεικτικές τιμές (αργότερα ενημερώνονται αναλόγως).

```
volatile int rate[10];                    // array to hold last ten IBI values
volatile unsigned long sampleCounter = 0;          // used to determine pulse timing
volatile unsigned long lastBeatTime = 0;           // used to find IBI
volatile int P =512;                      // used to find peak in pulse wave, seeded
volatile int T = 512;                     // used to find trough in pulse wave, seeded
volatile int thresh = 525;                // used to find instant moment of heart beat, seeded
volatile int amp = 100;                   // used to hold amplitude of pulse waveform, seeded
volatile boolean firstBeat = true;        // used to seed rate array so we startup with reasonable BPM
volatile boolean secondBeat = false;      // used to seed rate array so we startup with reasonable BPM
```

Τέλος για να καταλάβουμε αν πρόκειται για τον 1ο παλμό ή όχι χρησιμοποιούμε 2 μεταβλητές, τις firstBeat και secondBeat.

Με τον παρακάτω κώδικα ορίζουμε τον χρόνο διακοπής κατά τον οποίο θα τρέχει η ρουτίνα και θα επεξεργάζεται τα δεδομένα.

```
void interruptSetup(){
  // Initializes Timer2 to throw an interrupt every 2mS.
  TCCR2A = 0x02;     // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE
  TCCR2B = 0x06;     // DON'T FORCE COMPARE, 256 PRESCALER
  OCR2A = 0XFF;      // SET THE TOP OF THE COUNT TO 255 FOR 250Hz SAMPLE RATE
  TIMSK2 = 0x02;     // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A
  sei();             // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}
```

Κάθε φορά που ο μετρητής του arduino φτάνει στο 255 (αριθμός που υπολογίζεται σύμφωνα με τον oscillator του μικροεπεξεργαστή ούτως ώστε να έχουμε διακοπή κάθε 4ms), διαβάζουμε τα δεδομένα εισόδου από το pulsePin, και αυξάνουμε την μεταβλητή sampleCounter που κρατά τις στιγμές δειγματοληψίας. Να σημειωθεί ότι αφού έχουμε ορίσει το prescaler στα 256, το clock του συστήματος είναι πια 62.5 kHz. Ορίζουμε την μεταβλητή N που περιέχει τον χρόνο που πέρασε από την τελευταία φορά που είχαμε παλμό σε σχέση με την παρούσα στιγμή. Αρχικά παίρνει την τιμή 4.

```
ISR(TIMER2_COMPA_vect){                         // triggered when Timer2 counts to 255
  cli();                                        // disable interrupts while we do this
  Signal = analogRead(pulsePin);                // read the Pulse Sensor
  sampleCounter += 4;                           // keep track of the time in mS with this variable
  int N = sampleCounter - lastBeatTime;         // monitor the time since the last beat to avoid noise
```

Ακολουθεί ένας απλός κώδικας εύρεσης των min και max τιμών από όλα τα δεδομένα που λαμβάνει η μεταβλητή Signal. Ένας επιπλέον σημαντικός περιορισμός που βάζουμε είναι ότι πρέπει να έχουνε περάσει τα 3/5 του χρόνου από την τελευταία φορά που είχαμε παλμό προκειμένου να αποφύγουμε να πιαστεί η στιγμή που η τιμή θα είναι μεν μεγαλύτερη του threshold αλλά οφείλεται στις ανακλάσεις, να έχει περάσει δηλαδή η στιγμή του dichrotic notch για να έχουμε σωστές μετρήσεις.

```
    //  find the peak and trough of the pulse wave
  if(Signal < thresh && N > (IBI/5)*3){       // avoid dichrotic noise by waiting 3/5 of last IBI
    if (Signal < T){                          // T is the trough
      T = Signal;                             // keep track of lowest point in pulse wave
    }
  }

  if(Signal > thresh && Signal > P){          // thresh condition helps avoid noise
    P = Signal;                               // P is the peak
  }                                           // keep track of highest point in pulse wave
```

Για τα πρώτα 500ms όταν τρέχει η ρουτίνα διακοπής εκτελούνται μόνο οι παραπάνω γραμμές κώδικα για να βρούμε μια αρκετά καλή προσέγγιση των P και T στην αρχή.

Ο παρακάτω κώδικας εκτελείται αφού ελεγχθεί η συνθήκη N>500 που αναφέραμε. Όταν στην δεδομένη στιγμή που τρέχει η τιμή εισόδου είναι μεγαλύτερη του threshold που ορίσαμε και στην αμέσως προηγούμενη χρονική στιγμή που έτρεξε η ρουτίνα διακοπής δεν είχε ανιχνευτεί παλμός και έχουν περάσει τα 3/5 του IBI, βρίσκουμε παλμό! Μεταβάλουμε την κατάσταση της μεταβλητής Pulse true και έπειτα υπολογίζουμε τον νέο χρόνο IBI και lastBeatTime ο οποίος παίρνει ως τιμή την δεδομένη χρονική στιγμή.

```
if (N > 500){                                   // avoid high frequency noise
  if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) ){
    Pulse = true;                               // set the Pulse flag when we think there is a pulse
    IBI = sampleCounter - lastBeatTime;         // measure time between beats in mS
    lastBeatTime = sampleCounter;               // keep track of time for next pulse
```

Εφόσον έχουμε αρχικοποιήσει τον fistBeat με true και τον secondBeat με false, την πρώτη φόρα τρέχει ο κώδικας από την συνθήκη firstBeat όπου απλά πλέον αντιστρέφει τις τιμές των μεταβλήτων fistBeat, secondBeat και με το return επιστρέφει στην ρουτίνα CardioGraph. Την δεύτερη φορά που θα βρεθεί παλμός μπαίνουμε στην συνθήκη secondBeat όπου αρχικοποιείται ο πίνακας rate με την ίδια αρχική τιμή IBI καθώς δεν έχουμε 10 παλιότερες προφανώς και επίσης η μεταβλητή secondBeat γίνεται false για να μην ξαναεκτελεστεί κάποια από τις 2 παρακάτω.

```
if(secondBeat){                      // if this is the second beat, if secondBeat == TRUE
  secondBeat = false;                // clear secondBeat flag
  for(int i=0; i<=9; i++){           // seed the running total to get a realisitic BPM at startup
    rate[i] = IBI;
  }
}

if(firstBeat){                       // if it's the first time we found a beat, if firstBeat == TRUE
  firstBeat = false;                 // clear firstBeat flag
  secondBeat = true;                 // set the second beat flag
  sei();                             // enable interrupts again
  return;                            // IBI value is unreliable so discard it
}
```

Στην περίπτωση που είμαστε στον 3ο παλμό που βρέθηκε και μετά, πλέον τρέχει και ο παρακάτω κώδικας όπου κάθε φορά «διαγράφει» την πιο παλιά τιμή IBI του πίνακα rate, και προσθέτει την καινούρια που υπολογίστηκε πιο πάνω. Επίσης βρίσκουμε και το BPM σύμφωνα με τον πίνακα rate. Τέλος θέτουμε την μεταβλητή QS με true για να δείξουμε στην εξωτερική ρουτίνα ότι βρέθηκε παλμός.

```
// keep a running total of the last 10 IBI values
word runningTotal = 0;                   // clear the runningTotal variable

for(int i=0; i<=8; i++){                 // shift data in the rate array
  rate[i] = rate[i+1];                   // and drop the oldest IBI value
  runningTotal += rate[i];               // add up the 9 oldest IBI values
}

rate[9] = IBI;                           // add the latest IBI to the rate array
runningTotal += rate[9];                 // add the latest IBI to runningTotal
runningTotal /= 10;                      // average the last 10 IBI values
BPM = 60000/runningTotal;                // how many beats can fit into a minute? that's BPM!
QS = true;                               // set Quantified Self flag
// QS FLAG IS NOT CLEARED INSIDE THIS ISR
```

Για την περίπτωση που το δεδομένο στο Signal είναι μικρότερο του threshold και προηγουμένως είχαμε παλμό, θέτουμε την μεταβλητή Pulse στην τιμή false, και υπολογίζουμε το νέο πλάτος σύμφωνα με τις τελευταίες μετρήσεις του P και T που μπορεί να έχουν καλύτερη προσέγγιση στην αληθινή τιμή πλέον. Επίσης ορίζουμε το νέο threshold στο 50% του νέου πλάτους και επαναπροσδιορίζουμε τα P και T σύμφωνα με την μεταβλητή thesh.

```
if (Signal < thresh && Pulse == true){   // when the values are going down, the beat is over
  Pulse = false;                         // reset the Pulse flag so we can do it again
  amp = P - T;                           // get amplitude of the pulse wave
  thresh = amp/2 + T;                    // set thresh at 50% of the amplitude
  P = thresh;                            // reset these for next time
  T = thresh;
}
```

Τέλος, για την περίπτωση που έχουν περάσει 2.5 seconds από την τελευταία καταμέτρηση παλμού αρχικοποιούμε τις μεταβλητές όπως στην αρχή του προγράμματος για να γίνει ουσιαστικά μια επανάληψη της διαδικασίας από την αρχή.

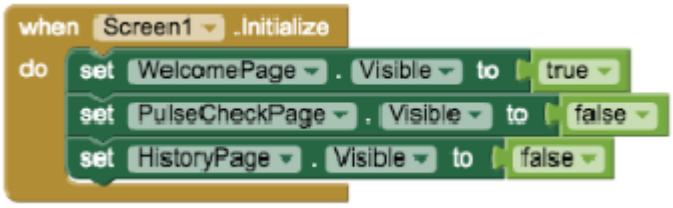
```
if (N > 2500){                          // if 2.5 seconds go by without a beat
  thresh = 512;                         // set thresh default
  P = 512;                              // set P default
  T = 512;                              // set T default
  lastBeatTime = sampleCounter;         // bring the lastBeatTime up to date
  firstBeat = true;                     // set these to avoid noise
  secondBeat = false;                   // when we get the heartbeat back
}

sei();                                  // enable interrupts when youre done!
```

4. Εφαρμογή σε Android

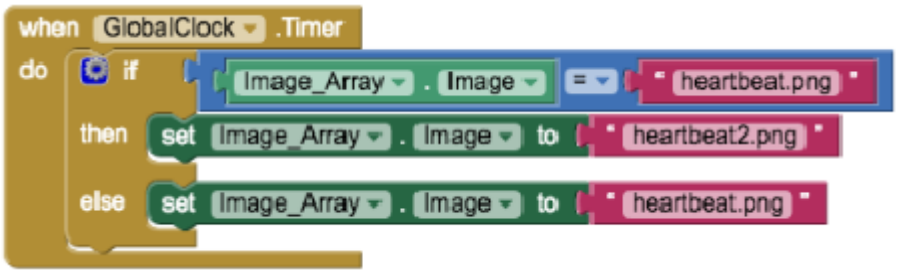
Για την δημιουργία της εφαρμογής σε Android χρησιμοποιήσαμε το λογισμικό App Inventor 2, ένα cloud-based εργαλείο που βοηθά στην κατασκευή εφαρμογών από το διαδίκτυο. Χρησιμοποιούνται έτοιμα blocks με εντολές τα οποία συνδέεις μεταξύ τους και η «γλώσσα» μοιάζει με την Java. Στην εφαρμογή χρησιμοποιούμε μια tinyDB που αποθηκεύει τις 5 τελευταίες μετρήσεις BPM του χρήστη.

4.1 Κώδικας με το App Inventor2

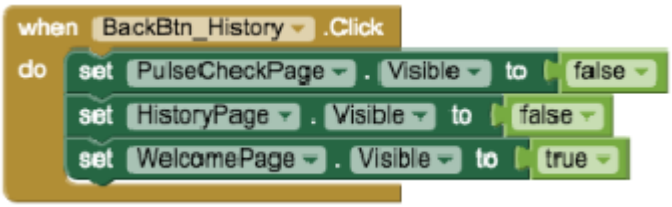
Σε αυτό το block εντολών ορίζουμε ποια σελίδα θα εμφανίζεται αρχικά όταν ανοίγει η εφαρμογή. Αρχικοποιούμε την Welcome Page.



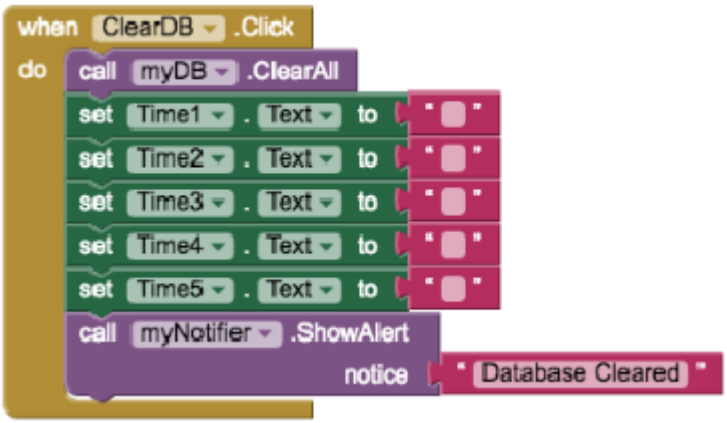
Σε αυτό το block εντολών δημιουργούμε το logo με την καρδιά που αλλάζει χρώμα στην Welcome Page σύμφωνα με το ρολόι GlobalClock.



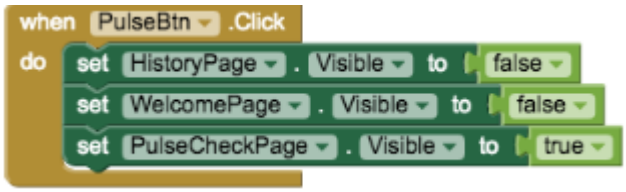
Όταν είμαστε στη σελίδα History και πατηθεί το κουμπί Back μας επιστρέφει στην Welcome Page.



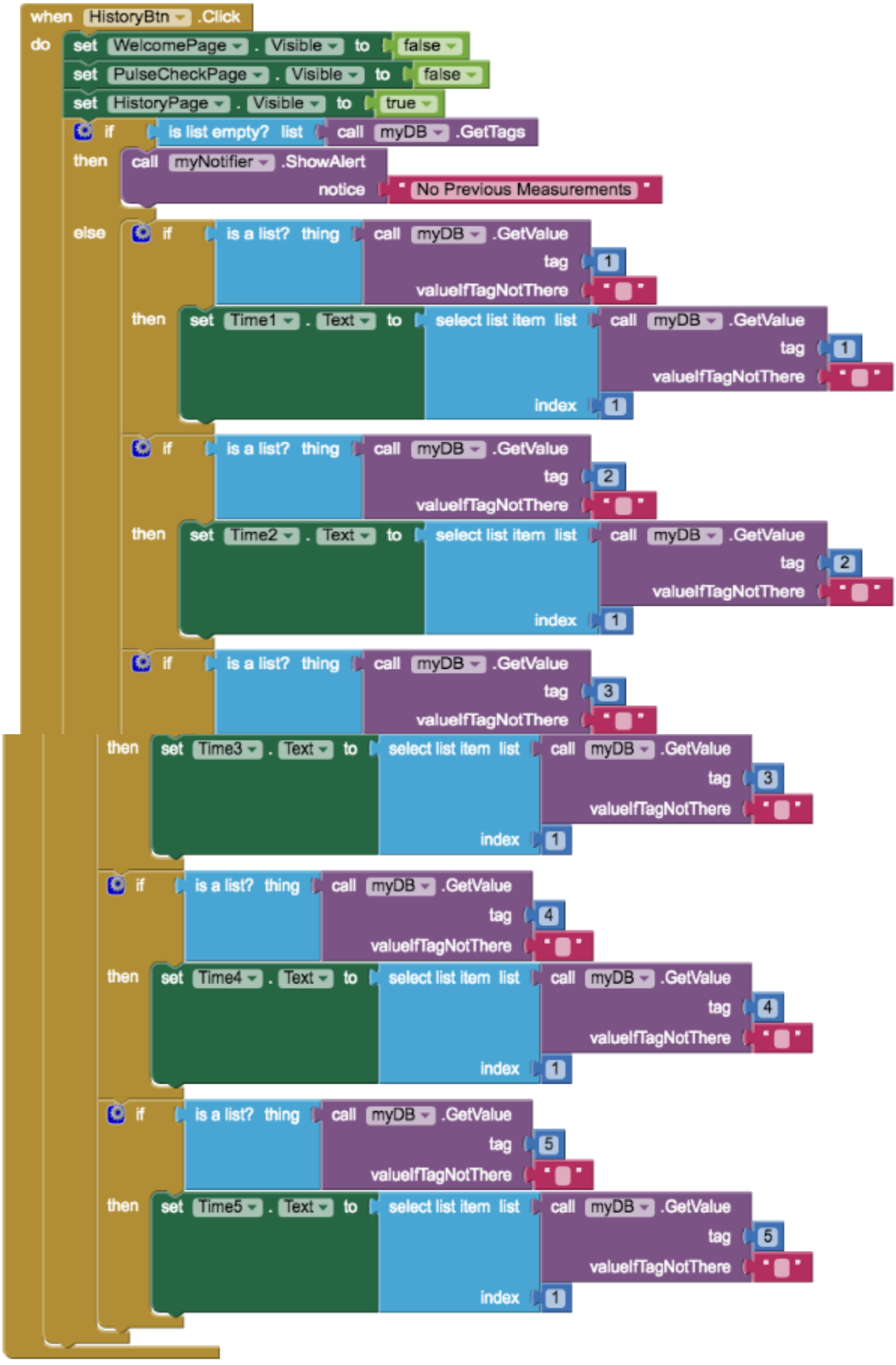
Όταν πατάμε το κουμπί ClearDatabase στην HistoryPage καλείται η συνάρτηση clearAll που καθαρίζει την tinyDB που χρησιμοποιούμε για την αποθήκευση των 5 τελευταίων τιμών BPM του χρήστη και στη θέση των τιμών στις ετικέτες Time1-Time5 μπαίνει το κενό. Τέλος εμφανίζεται μήνυμα επιτυχούς διαγραφής της βάσης δεδομένων.



Όταν είμαστε στην αρχική σελίδα,Welcome Page, και πατηθεί το κουμπί Check my Pulse, μεταβαίνουμε στην σελίδα PulseCheckPage.



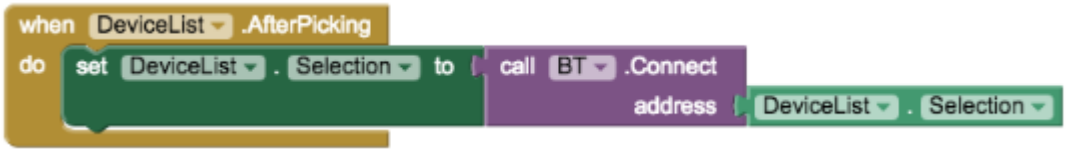
Όταν είμαστε στην αρχική σελίδα και πατηθεί το κουμπί History, μεταβαίνουμε στην σελίδα HistoryPage. Με κατάλληλη εντολή καλούμε τα δεδομένα που έχουν αποθηκευτεί στην βάση δεδομένων, αν δεν υπάρχουν τότε εμφανίζεται το μήνυμα “No Previous Measurements”. Όταν υπάρχουν αποθηκευμένα δεδομένα εμφανίζονται δίπλα από τις ετικέτες Time1, Time2, Time3, Time4, Time5(ημέρα και ώρα) οι αντίστοιχες τιμές των BPM.



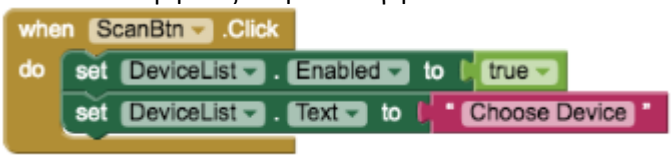
Όταν έχουμε πατήσει το κουμπί Choose Device καλείται αυτή η ρουτίνα για να εμφανίσει τα ονόματα των διαθέσιμων devices για σύνδεση μέσω Bluetooth.

Όταν είμαστε στην σελίδα PulseCheckPage και πατάμε το κουμπί Back, επιστρέφουμε στην αρχική σελίδα Welcome Page.

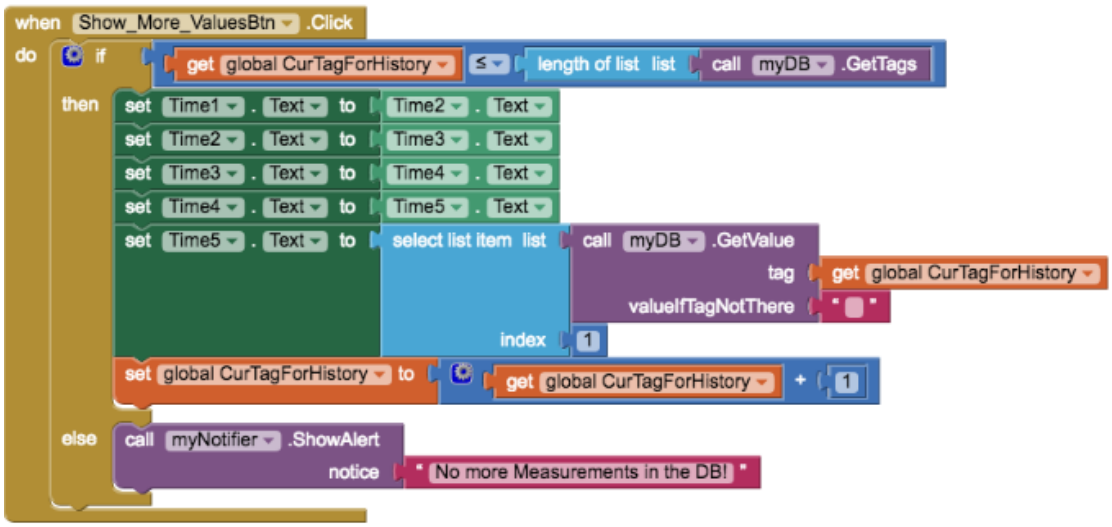
Όταν έχουμε πατήσει το κουμπί Choose Device και έχουν επιλέξει από τις διαθέσιμες συσκευές που εμφανίστηκαν τρέχει ο παρακάτω κώδικας και συνδέεται με την συσκευή που επιλέξαμε για την μεταφορά δεδομένων.



Όταν είμαστε στην σελίδα checkPulsePage και πατάμε το κουμπί Scan ενεργοποιεί το Bluetooth του Arduino και εμφανίζει την επιλογή Choose Device.

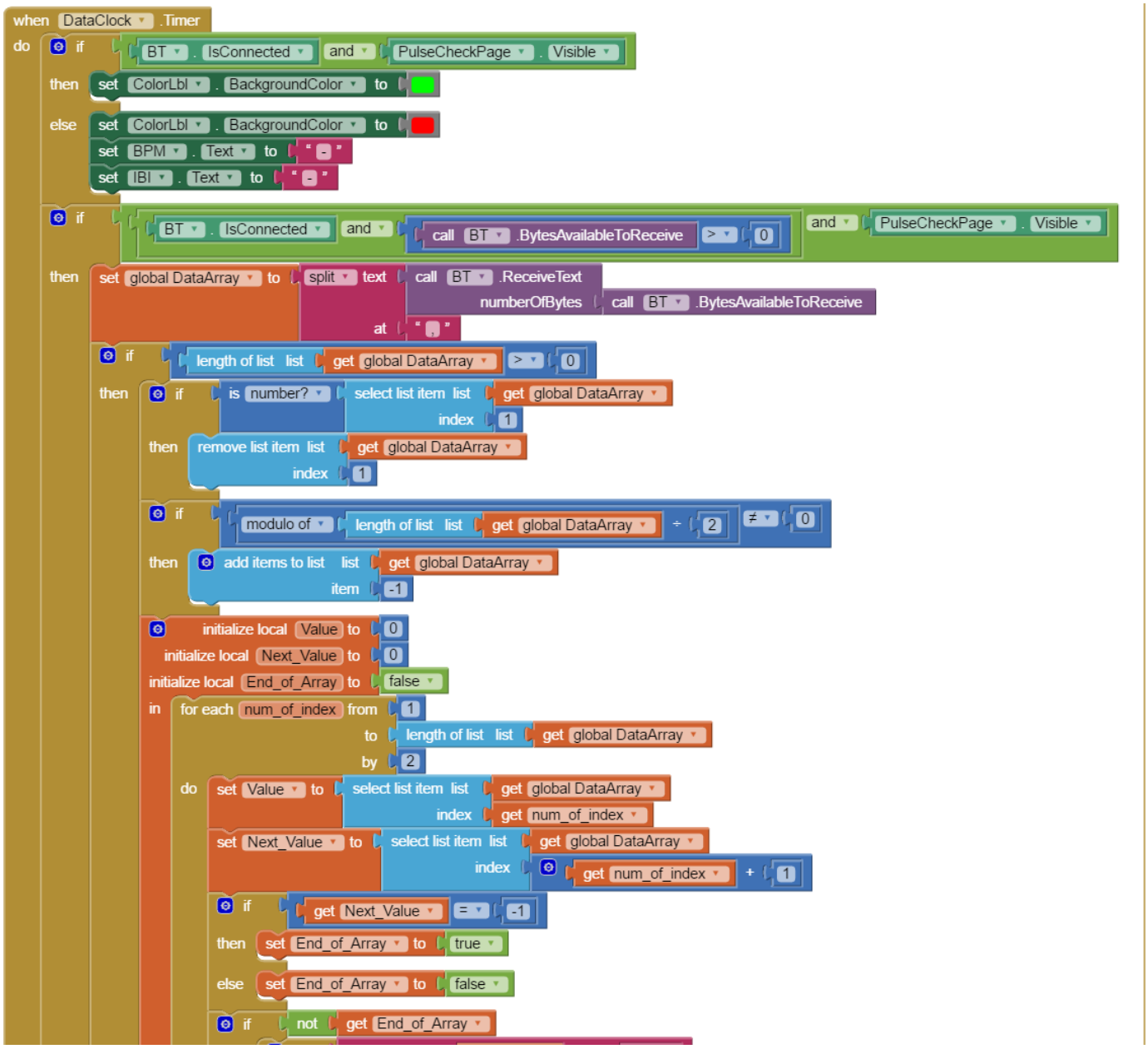


Όταν πατάμε το κουμπί Show More Measurements από την HistoryPage, αν υπάρχουν παραπάνω από 5 τιμές στην DB ανασύρονται και εμφανίζονται στην σελίδα στην θέση των παλαιότερων 5 στις ετικέτες Time1-5. Αλλιώς εμφανίζεται μήνυμα ότι δεν υπάρχουν άλλες μετρήσεις.

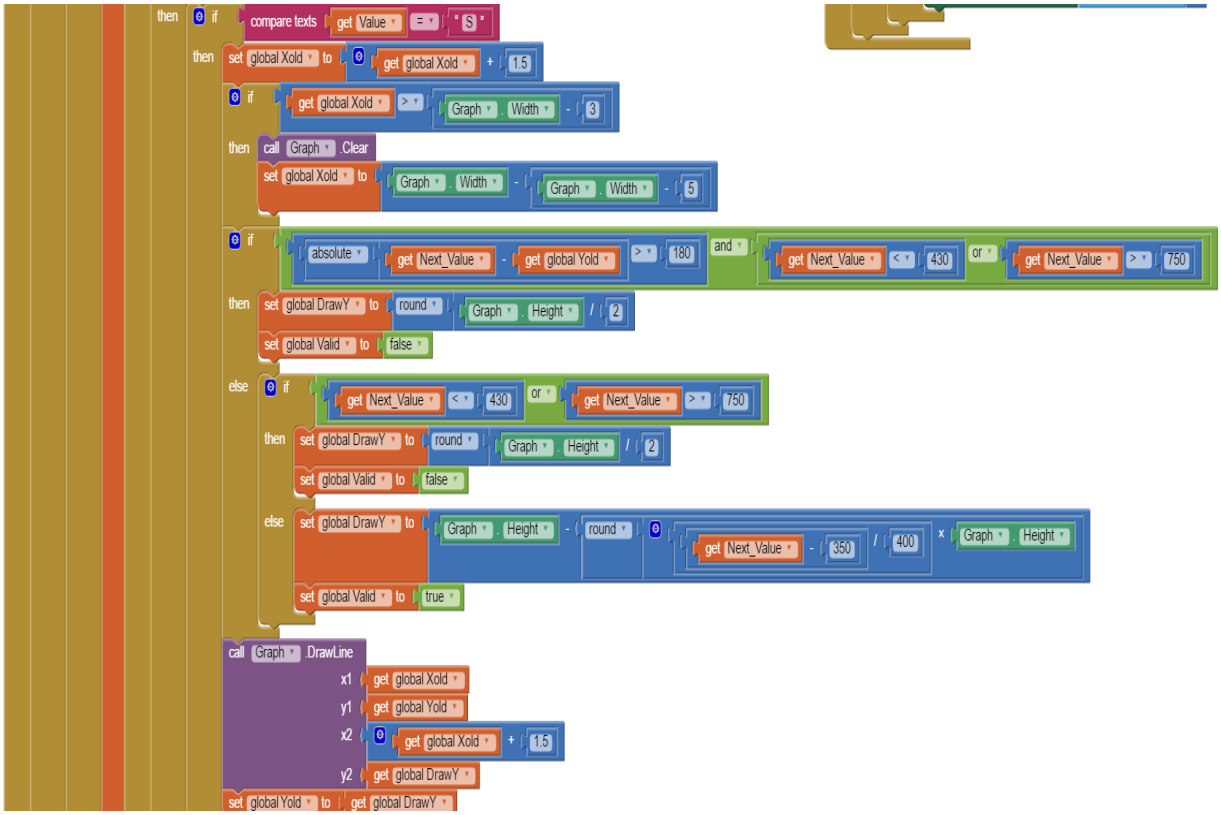


Αρχικοποιούμε μεταβλητές που σχετίζονται με το διάγραμμα και τις σημαίες.

Ανά 1 msec έχουμε ενεργοποιήσει το DataClock, το οποίο ελέγχει το Bluetooth για την άφιξη δεδομένων. Όταν υπάρχουν διαθέσιμα Bytes για λήψη, χωρίζουμε τα δεδομένα με split (τα δεδομένα διαχωρίζονται με κόμμα από την ρουτίνα Cardiograph). Προσέχουμε να μην έχουμε ως πρώτο στοιχείο αριθμό, διότι μετά έχουμε απροσδιοριστία στο τι ακολουθεί.

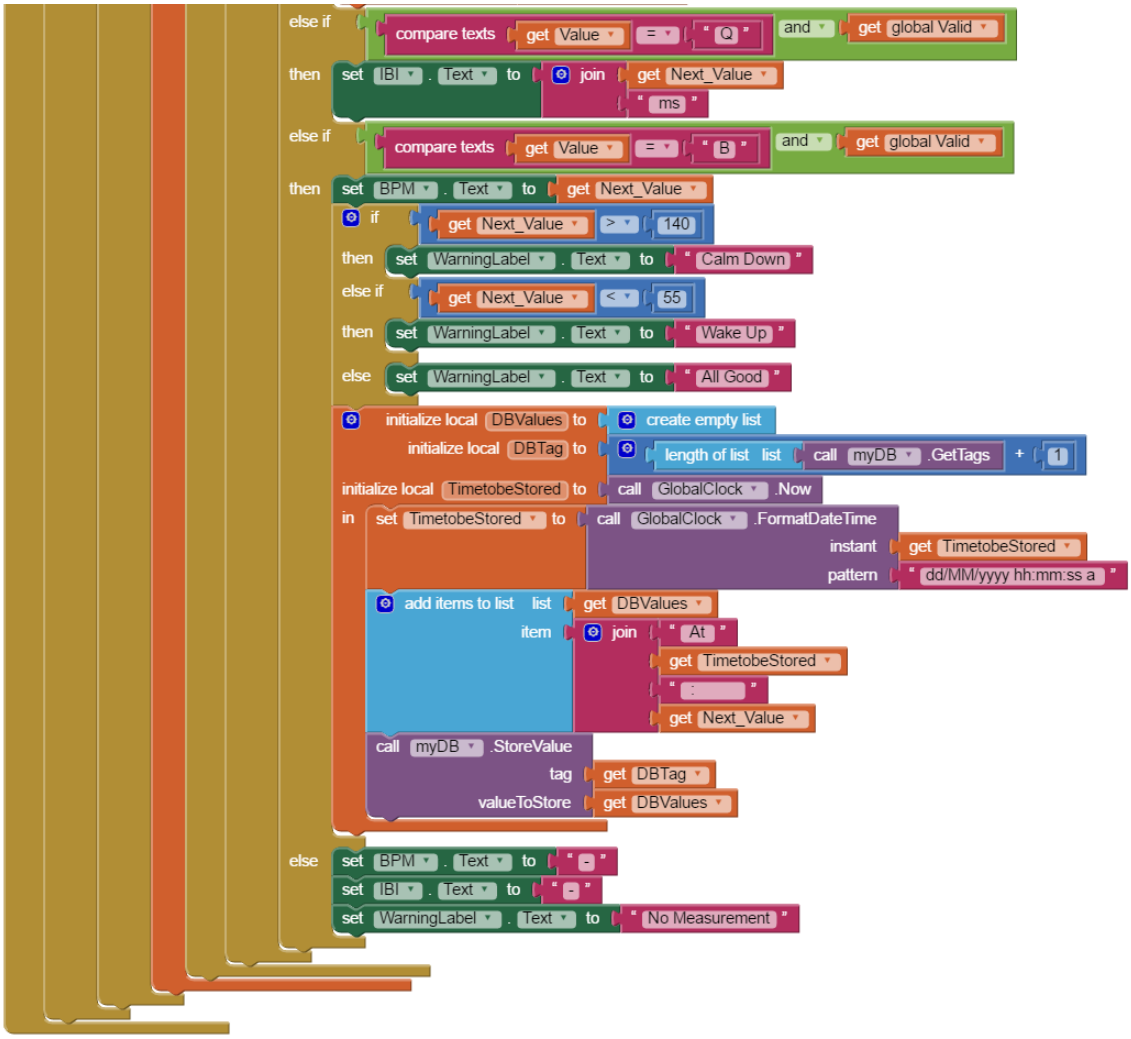


Για τιμές που αρχίζουν από S, σχηματίζουμε το γράφημα. Ελέγχουμε ώστε να μην ξεπερνάμε τα όρια της εικόνας (αλλιώς διαγράφουμε το προηγούμενο γράφημα και ξεκινάμε πάλι από την αρχή – την οποία θεωρούμε στο 5). Ύστερα κάνουμε κάποιους ελέγχους ώστε να μειωθούν οι θόρυβοι που απεικονίζονται στο γράφημα. Επειδή η τιμές που περιμένουμε βρίσκονται γύρω στο 512, αποκλείουμε τιμές κάτω από 430 και πάνω από 750. Ακόμα, αν μεταξύ διαδοχικών τιμών υπάρχει διαφορά μεγαλύτερη από 180, θεωρούμε ότι οφείλεται σε θόρυβο. Μόλις εντοπίσουμε αυτές τις καταστάσεις, εμφανίζουμε μια flat line για να δείξουμε ότι δεν πρόκειται για πραγματικές μετρήσεις.



Όλες οι υπόλοιπες τιμές εμφανίζονται με αλλαγή κλίμακας για μεγαλύτερη ευκρίνεια.

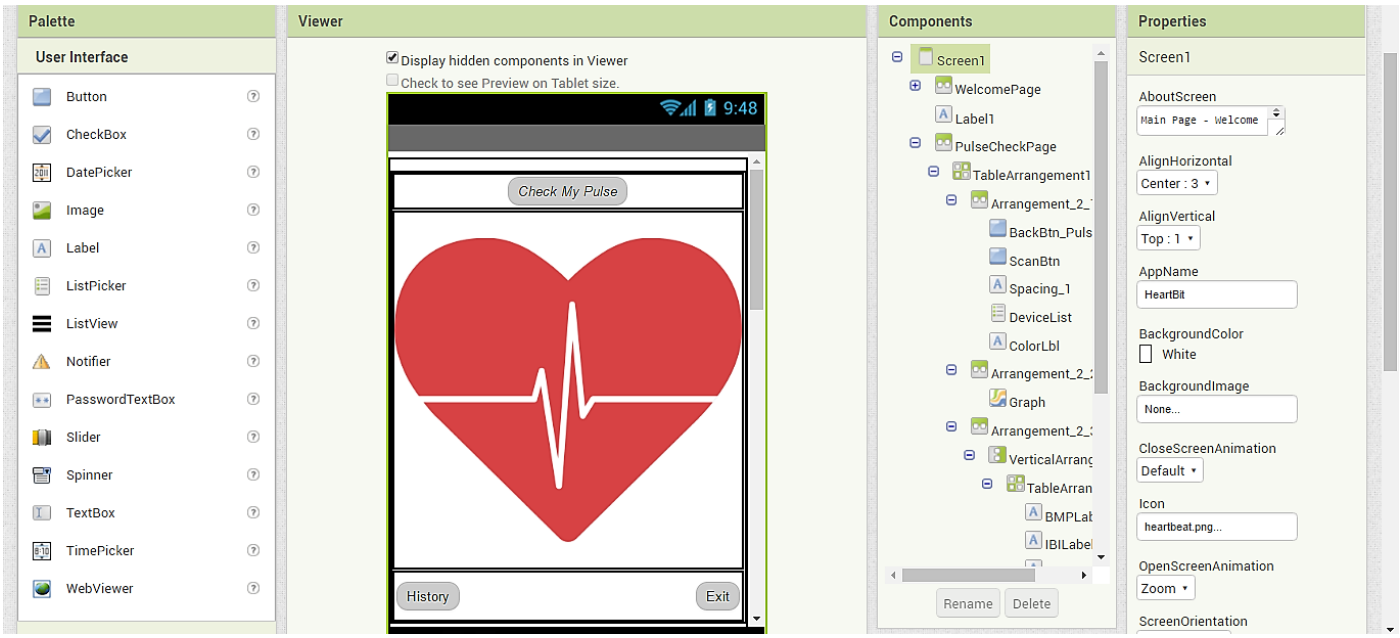
Εμφανίζουμε τις τιμές BPM και IBI, καθώς και προειδοποιητικά μηνύματα στην οθόνη όταν ο σφυγμός είναι είτε πολύ υψηλός είτε πολύ χαμηλός. Τέλος, τοποθετούμε την τιμή BPM στην tinyDB μαζί με τον χρόνο (ημερομηνία και hh:ss:mm). Όταν δεν λαμβάνουμε στοιχεία το γράφημα είναι άδειο και δεν εμφανίζονται τιμές.



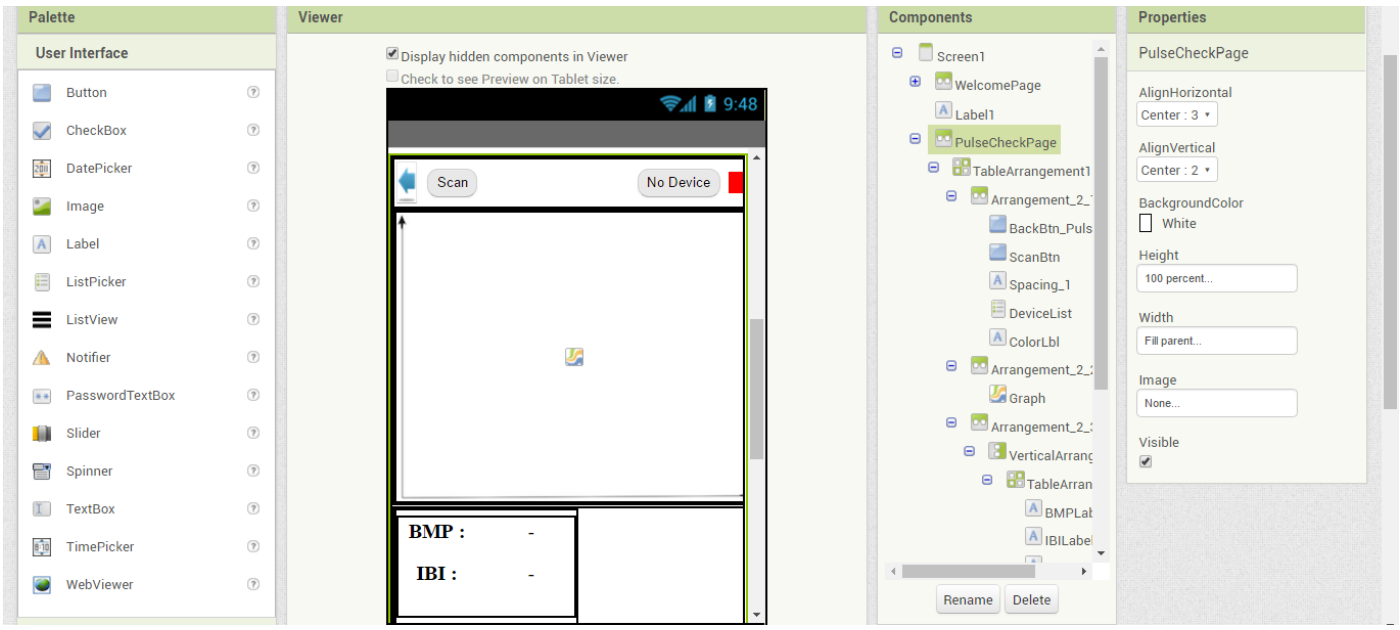
Όταν πατάμε το κουμπί Exit από την αρχική σελίδα, βγαίνουμε από την εφαρμογή.

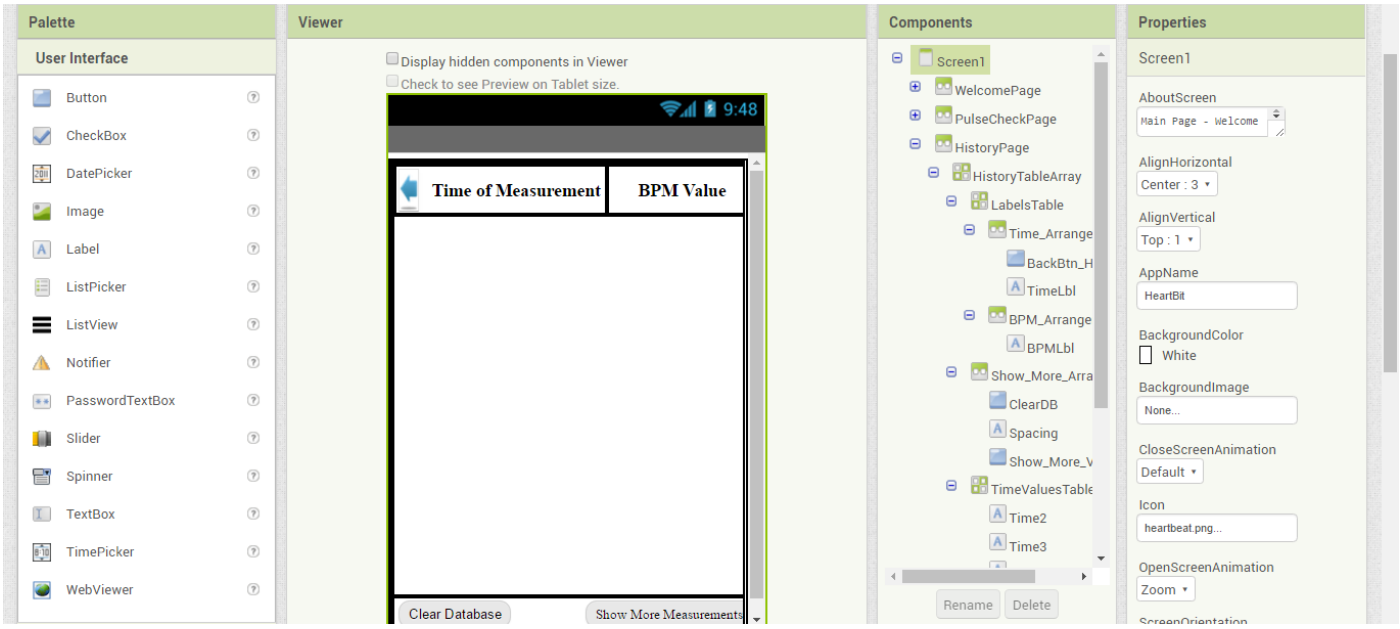
4.2 Η εφαρμογή σε Android

Αυτή είναι η αρχική σελίδα της εφαρμογής μας (WelcomePage). Ουσιαστικά πρόκειται για μία σελίδα αλλά η οποία έχει ορατά και μη ορατά χαρακτηριστικά ανάλογα τις επιλογές που θα κάνουμε. Αυτό ήταν απαραίτητο διότι η χρήση του Bluetooth Client με τις πολλαπλές σελίδες παρουσιάζει προβλήματα.



Παρακάτω φαίνονται οι σελίδες μέτρησης του καρδιακού παλμού και του ιστορικού με τις τελευταίες πέντε τιμές του BPM. Στον χώρο του γραφήματος σχηματίζεται ένα ενδεικτικό καρδιογράφημα.




Palette
User Interface
Button
CheckBox
DatePicker
Image
Label
ListPicker
ListView
Notifier
PasswordTextBox
Slider
Spinner
TextBox
TimePicker
WebViewer
Viewer
Display hidden components in Viewer
Check to see Preview on Tablet size.
9:48
Time of Measurement
BPM Value
Clear Database
Show More Measurements
Components
Screen1
WelcomePage
PulseCheckPage
HistoryPage
HistoryTableArray
LabelsTable
Time_Arrange
BackBtn_H
TimeLbl
BPM_Arrange
BPMLbl
Show_More_Arra
ClearDB
Spacing
Show_More_V
TimeValuesTable
Time2
Time3
Rename
Delete
Properties
Screen1
AboutScreen
Main Page - Welcome
AlignHorizontal
Center : 3
AlignVertical
Top : 1
AppName
HeartBit
BackgroundColor
White
BackgroundImage
None...
CloseScreenAnimation
Default
Icon
heartbeat.png...
OpenScreenAnimation
Zoom

5. Βιβλιογραφία και άλλες πηγές

[1]. https://github.com/WorldFamousElectronics/PulseSensor_Amped_Arduino

[2]. https://www.instructables.com/id/Arduino-Pulse-Sensor-Cardio-Graph/?ALLSTEPS