JCONES Vol. 8, No. 2 (2019) 87-94

Journal of Control and Network Systems

Situs Jurnal : http://jurnal.stikom.edu/index.php/jcone

EFISIENSI DAYA UNTUK PANTAUAN DATA *HEART RATE* MENGGUNAKAN METODE *IDLE TIME-DEEP SLEEP*

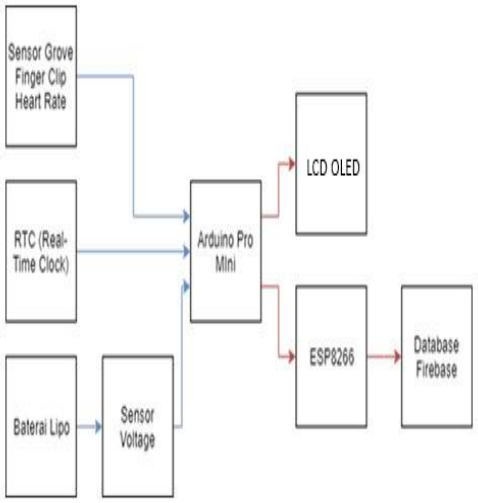
Yonanta Ferdinan Susanto [1] **Pauladie Susanto** [2] **Musayyanah**[3]
Program Studi/Jurusan Teknik Komputer
Institut Bisnis dan Informatika Stikom Surabaya
Jl. Raya Kedung Baruk 98 Surabaya, 60298s
Email: 1)15410200025@stikom.Edu 2)Pauladie@stikom.Edu, 3)musayyanah@stikom.Edu

Abstrak: Perkembangan teknologi saat ini sudah sangat berkembang pesat, terlebih pada teknologi yang bersifat IoT (*Internet of Things*). Suatu sistem Monitoring hardware yang bersifat *wearable* dan *IoT* serta hemat daya diperlukan dan dapat diterapkan untuk memudahkan proses pantauan. Sistem dan *Prototype* ini sebelumnya sudah di lakukan pada penelitian (Musayyanah, 2018) yang berdjudul Monitoring THR (Target Heart Rate) untuk optimalisasi latihan lari berbasis IoT (*Internet of Things*). Namun dalam penelitian ini terdapat kekurangan yaitu pada bagian konsumsi daya, maka dari itu algoritma Efisiensi *Idle-Time Deep Sleep* diimplementasikan pada penggabungan hardware sensor *Grove finger clip heart rate*, RTC (*Real-Time Clock*), modul Transmisi *Wireless* (Modul ESP8266) dan Arduino Pro Mini. Digunakan untuk melakukan pantauan data HR (*Heart Rate*) dan selanjutnya masuk *Database Firebase* untuk memudahkan pelatih melakukan pantauan. Dengan menggunakan algoritma tersebut, maka penghematan daya pada *Prototype* berhasil didapatkan. Ditunjukkan dengan hasil data durasi waktu pada fungsi LowPower.powerDown rata – rata berkisar 80,56 menit, untuk LowPower.idle rata – rata berkisar 73,3 menit sedangkan untuk Delay(1000) rata – rata berkisar 39,9 menit. *Prototype* tetap dapat berjalan dengan baik serta sesuai fungsi pada penelitian sebelumnya yang telah dilakukan.

Kata Kunci: *Internet Of Things*, Olahraga, *Heart Rate, Idle-Time Deep Sleep, LowPower.powerDown, LowPower.idle, Delay (1000).*

PENDAHULUAN

Teknologi IoT pada masa sekarang telah mengalami perkembangan yang pesat dan telah diterapkan di berbagai alat yang telah mendukung dan membutuhkan konsep IoT sehingga melahirkan wearable device, namun untuk kondisi sekarang peralatan tersebut masih belum bisa dikatakan sempurna, karena masih memiliki berbagai kekurangan yang harus diperbaiki, khususnya pada masalah konsumsi daya yang dibutuhkan untuk menjalankan berbagai alat yang telah diciptakan. maka dari itu diperlukan sebuah penelitian lanjutan agar alat-alat tersebut sangat ramah penggunaanya dan lebih mudah dalam menggunakannya. Namun untuk mengiplementasikan konsep IoT tidak semudah yang dibayangkan saat ini, karena dituntut untuk memiliki kemampuan Power Management yang baik (Martinez, et al., 2015). Implementasi teknologi IoT dapat digunakan pada perangkat dengan protokol yang mendukung IoT, seperti protokol Zigbee (Xbee), protokol Lora (Lora) Namun perangkat tersebut membutuhkan biaya yang besar untuk satu node transmisi. Implementasi IoT biasanya digunakan pada protokol Wifi (IEEE 802.11), contohnya pada modul Esp8266 atau Wemos, namun kedua perangkat tersebut mengkonsumsi daya yang besar, seperti yang telah dilakukan pada penelitian (Musayyanah, 2018). Penghematan daya pernah dilakukan pada (Anneke, 2018) yang berjudul "Monitoring Hasil Tembakau Berbasis IoT di Baristan Surabaya menggunakan metode *Idle Time - DeepSleep* untuk pantauan tanaman tembakau, dimana metode tersebut dapat menghemat baterai 75% sesuai dengan jadwal pantauan. Penerapan *wearable device* untuk deteksi sehingga penelitian ini menerapkan metode *Idle Time Deep Sleep* pada *device* pantauan data HR untuk penghematan daya baterai pengambilan data HR (*Heart Rate*).



Gambar 1. Diagram sistem keseluruhan

METODE PENELITIAN

Dibagian ini, pembahasan materi dibagi menjadi 2 seperti berikut:

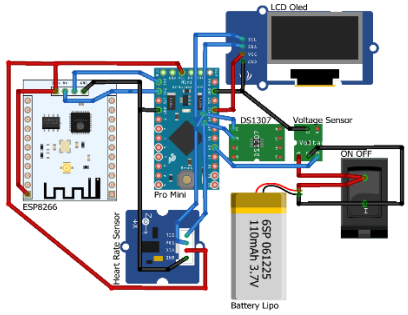
A. Perancangan Perangkat Keras

Dalam pembahasan kali ini, dijelaskan tentang blok diagram diatas, rancang bangun alat, pengolah data dan hasil dari rancang bangun. Berikut ini adalah penjelasannya:

Dalam blok diagram pada gambar 1 dibagi menjadi 3 bagian yaitu input, pengolah data, dan output. Input sendiri terdiri dari sensor *grove finger clip heart rate* untuk membaca data HR(*Heart Rate*), RTC (*Real – Time Clock*) digunakan untuk membaca data waktu dan memberi input data waktu pada mikrikontroler Arduino Pro Mini, baterai lipo sebagai catu daya seluruh sistem dan sensor *voltage* digunakan untuk membaca voltase baterai yang tersisa.

Pada bagian komponen pengolah data ada Arduino Pro Mini sebagai pengolahnya dan menerima beberapa data input yaitu dari sensor *grove finger clip heart rate* berupa nilai analog HR (*heart rate*) yang dikirim dengan cara serial, dan dikirimkan ke ESP8266 yang oleh ESP8266 dikirimkan ke database *Firebase*, untuk dilihat data dari *Prototype* efisiensi daya pantauan *heart rate.* Pada bagian output/keluaran *Prototype*, ditampilkan pada LCD OLED berupa data HR (Heart Rate) dan sisa baterai yang masih ada. Lalu mikrokontroller mengirimkan data ke ESP8266 untuk dikirim ke *Database Firebase*, namun data masih berupa data yang belum di pecah, tugas ESP8266 adalah memecah, sekaligus mengirimkan data ke Database *Firebase*.

2. Rangkaian *Prototype* Efisiensi Daya pantauan data Heart Rate.

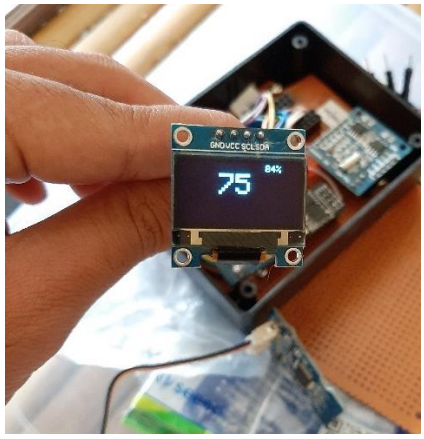


Gambar 2. Rangkaian *Prototype* efisiensi daya pantauan data *Heart Rate*.

Dalam rangkaian pada gambar 2 terdapat beberapa komponen *Prototype*, antara lain:

1. Sensor *Grove finger clip heart rate*
2. RTC (Real-Time Clock)
3. Sensor Voltage
4. Baterai lipo
5. Modul charger baterai lipo
6. Switch On Off
7. Arduino Pro Mini
8. Esp8266

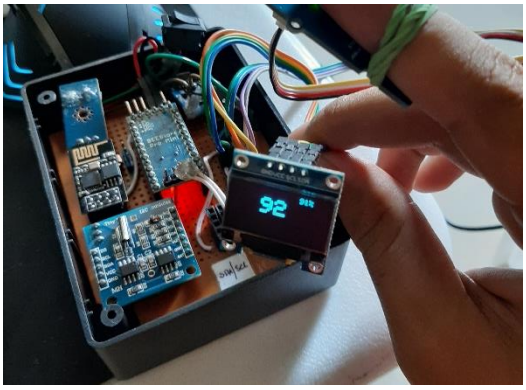
3. Tampilan display Lcd *Prototype*



Gambar 3. Tampilan display Lcd *Prototype*

Keterangan:

a. Nilai HR (*Heart Rate*)
   Digunakan untuk menampilkan data *Heart Rate* dari pembacaan sensor *grove finger clip heart rate*, yang ditampilkan pada LCD OLED.
b. Sisa Baterai dalam %
   Digunakan untuk memberikan informasi sisa daya baterai yang masih ada



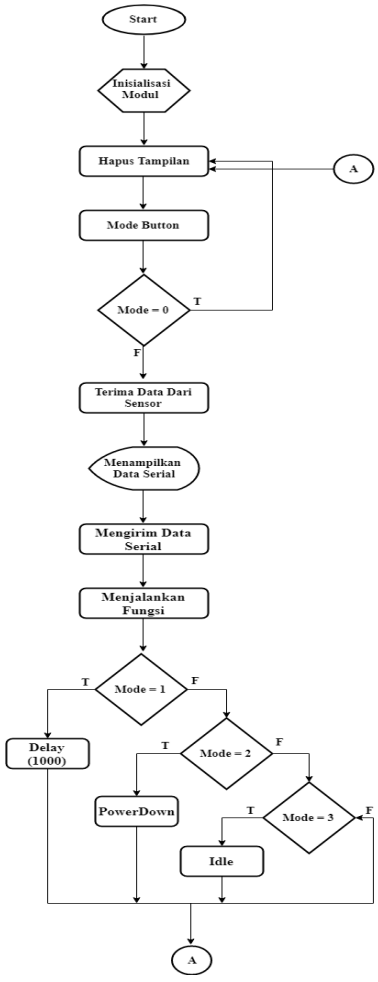
Gambar 4 : Tampilan Display LCD OLED Arduino

Karena dalam mengusung konsep IoT dan *wearable device*, *Prototype* yang dihasilkan juga memiliki dimensi yang kecil, serta menggunakan komponen yang berdimensi kecil juga ,sehingga memudahkan dalam pemakaian *Prototype* oleh olahragawan atau sekedar digunakan untuk memantau data HR (*Heart Rate*) penggunanya.

B. Perancangan Program Untuk Prototype

Pada bagian ini dijelaskan mengenai perancangan program yang dilakukan dan selanjutnya ditanamkan pada Arduino Pro Mini, dimana Arduino Pro Mini sendiri digunakan sebagai pusat pengolahan data yang didapat dari sensor pada *Prototype*. Berikut merupakan penjelasannya.

1. Pengolahan data pada Arduino Pro Mini



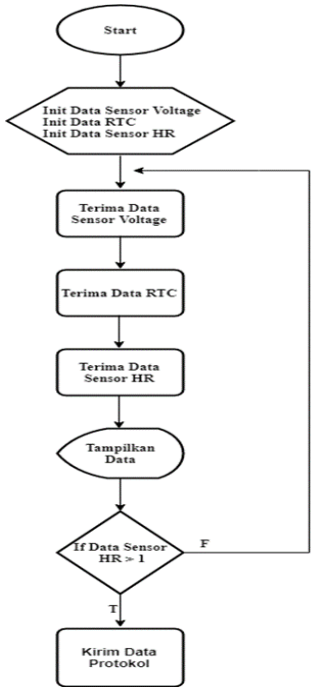
Gambar 5. Flowchart Arduino Pro Mini

Pada Gambar 5 *flowchart* yang dibuat sudah sangat jelas, dalam *flowchart* pada gambar 5 ditunjukkan bahwa ketika Arduino Pro Mini dalam posisi menyala, maka Arduino Pro Mini langsung melakukan proses inisialisasi modul yang tersambung pada Arduino Pro Mini, selanjutnya Arduino mengalami proses menerima data dari modul *sensor grove finger clip heart rate*, sensor voltage dan RTC (*Real-Time Clock*).

Setelah itu Arduino Pro Mini menampilkan data serial tersebut melalui komunikasi *i2c* ke LCD OLED, lalu Arduino Pro Mini mengirimkan data serial ke komponen ESP8266 untuk dikirimkan ke database *Firebase* setelah itu menjalankan fungsi algoritma efisiensi.

2. Terima Data Dari Sensor dan RTC

Pada bagian ini dijelaskan tentang proses terima data Arduino yang didapatkan dari modul sensor dan modul RTC, dimana alurnya adalah Arduino menginisialisasi semua modul yang terhubung dengan Arduino, kemudian jika sudah melakukan proses tersebut lalu Arduino terima data dari masing-masing komponen modul sensor dan modul RTC, kemudian menampilkan data ke dalam monitor LCD OLED dengan koneksi *i2c*, kemdian Arduino melakukan percabangan data, dimana ketika data sensor sudah melebihi nilai 1, maka Arduino mengirim data ke ESP8266 sebagai data yang dikirim ke database *Firebase*, jika belum memenuhi syarat nilai, maka Arduino melakukan proses terima data dari masing–masing komponen modul sensor dan modul RTC. Berikut adalah flowchart nya.

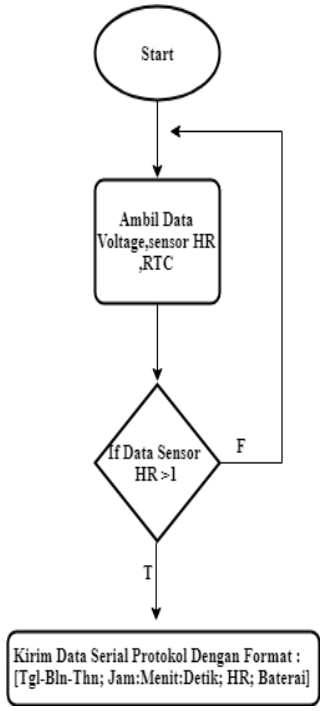


Gambar 6. *Flowchart* terima data dari sensor dan RTC

Flowchart ini menjelaskan alur pengiriman data serial, mulai dari menerima data dari masing–masing komponen input lalu masuk ke Arduino Pro Mini untuk diolah datanya, kemudian ditampilkan pada LCD OLED, dan untuk data sensor *Grove finger clip heart rate* ketika nilai masih kurang dari 1 atau sama dengan 1 maka Arduino belum mengirimkan data ke ESP8266, ketika data sudah lebih dari 1 nilainya, maka Arduino mengirim data protokol/format bentuk data yang pengirimannya sudah ditentukan.

3. Kirim Data Protokol/format data

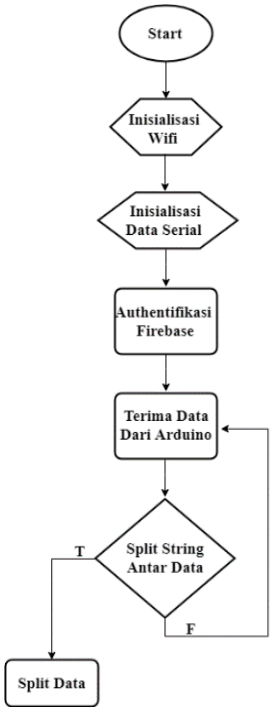
Kirim data protokol dijelaskan dengan *flowchart*, bagaimana sistem Arduino mengirim data protokol ke modul ESP 8266. Berikut adalah *flowchartnya*:



Gambar 7. Kirim data protokol

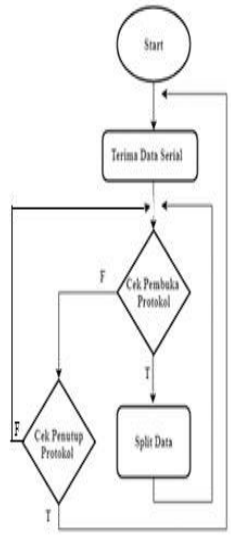
Pada gambar 7 dijelaskan tentang *flowchart* kirim data protokol ke ESP8266, Arduino mengirim data ini ke ESP8266 dalam komunikasi serial, lalu ESP 8266 membaca dan memecah datanya untuk dikirimkan ke Database *Firebase*, mula–mula mikrokontroller Arduino ambil/menerima data dari sensor voltage, sensor HR dan modul RTC, kemudian jika data sensor lebih dari 1 maka mengirim data protokol dengan format [Tgl-Bln-Thn; Jam:Menit:Detik;HR;Baterai]. Namun jika nilai dari sensor HR kurang dari 1 maka, sistem looping kembali untuk kembali ambil dan terima data dari sensor voltage, sensor HR dan Modul RTC dan melakukan proses seperti yang dijelaskan hingga data sensor bernilai > 1.

4. Pengolahan data pada ESP8266



Gambar 8. *Flowchart* ESP8266

Pada gambar 8 *flowchart* yang dibuat sudah sangat jelas, dalam *flowchart* pada gambar 6 ditunjukkan bahwa ketika ESP8266 dalam keadaan menyala maka ESP8266 langsung melakukan inisialisasi SSID yang sudah ada dalam program ESP8266, kemudian terkoneksi melalui jaringan wifi, karena dalam *Prototype* ini hasil data dari sensor ditampilkan pada database *Firebase*, maka selanjutnya ESP8266 melakukan authentifikasi *Firebase*, proses ini dilakukan agar *Prototype* bisa terkoneksi dengan database *Firebase*. Kemudian langkah selanjutnya adalah melakukan inisialisasi data yang diterima dari Arduino Pro Mini, setelah inisialisasi selesai maka ESP8266 menerima data dari Arduino Pro Mini. Setelah menerima data ESP melakukan proses pemecahan data string, proses ini dilakukan untuk menata data yang nantinya dikirimkan ke Database *Firebase* agar tampilan data di Database *Firebase* lebih tertata rapi. Setelah proses *split string* maka ESP8266 mengirimkan data ke Database *Firebase* menggunakan koneksi wifi.



Gambar 9. *Flowchart* Esp8266 terima data

Flowchart pada gambar 9 menjelaskan tentang ESP8266 menerima data serial yang dikirimkan dari mikrokontroler Arduino. Kemudian melakukan cek data pembuka protokol berupa string "[" jika data yang pertama dikirimkan adalah string "[" maka kondisi 'True" jika benar maka kemudian ESP8266 melakukan split data untuk kemudian mengirim ke Database *Firebase*, namun jika data yang dikirimkan kondisi nya "False" maka ESP8266 mengecek penutup protokol berupa data string "]" jika sudah sampai mengirim String penutup, maka ESP8266 kembali melakukan *looping* terima data selanjutnya dari Arduino Pro Mini untuk diolah kembali atau dipecah datanya kemudian dicek kembali dan selanjutnya dikirimkan ke database *Firebase*. Jika sudah melakukan pengiriman maka bisa dilihat pengamatan data pada database *Firebase*. Dilihat apakah data terkirim dengan baik atau masih tidak sesuai dengan format yang sudah diterapkan sebelumnya pada *split* data pada komponen ESP 8266. Jika sudah sesuai maka pengiriman bisa dikatakan berhasil.

HASIL DAN PEMBAHASAN

Dalam bagian pembahasan kali ini dijelaskan mengenai perbedaan efisiensi dari ke 3 fungsi yang digunakan untuk mendukung konsep efisiensi daya pada *Prototype* pantauan data heart rate.

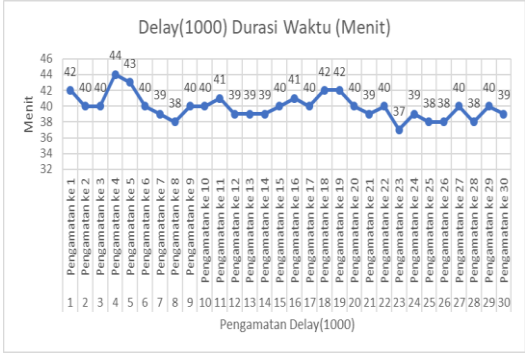
A. Pengujian Fungsi Delay(1000)

Pada pengujian kali ini dimaksudkan untuk mengetahui apakah dengan fungsi yang digunakan yaitu fungsi Delay(1000) bisa melakukan efisiesi dengan baik atau tidak. Dengan diterapkan fungsi ini pada *Prototype* membuat *Prototype* efisien atau tidak dalam mengkonsumsi daya. Hasil dari pengujian ditampilkan dalam bentuk tabel dan grafik agar bisa dilihat secara jelas.

Tabel 1. Hasil pengamatan durasi mengguakan fungsi Delay(1000)

No	Delay (1000)		
1.	Pengamatan ke 1 = 42 Menit	16.	Pengamatan ke 16 = 41 Menit
2.	Pengamatan ke 2 = 40 Menit	17.	Pengamatan ke 17 = 40 Menit
3.	Pengamatan ke 3 = 40 Menit	18.	Pengamatan ke 18 = 42 Menit
4	Pengamatan ke 4 = 44 Menit	19.	Pengamatan ke 19 = 42 Menit
5.	Pengamatan ke 5 = 43 Menit	20.	Pengamatan ke 20 = 40 Menit
6.	Pengamatan ke 6 = 40 Menit	21.	Pengamatan ke 21 = 39 Menit
7.	Pengamatan ke 7 = 39 Menit	22.	Pengamatan ke 22 = 40 Menit
8.	Pengamatan ke 8 = 38 Menit	23.	Pengamatan ke 23 = 37 menit
9.	Pengamatan ke 9 = 40 Menit	24.	Pengamatan ke 24 = 39 Menit
10.	Pengamatan ke 10 = 40 Menit	25.	Pengamatan ke 25 = 38 Menit
11.	Pengamatan ke 11 = 41 Menit	26.	Pengamatan ke 26 = 38 Menit
12.	Pengamatan ke 12 = 39 Menit	27.	Pengamatan ke 27 = 40 Menit
13.	Pengamatan ke 13 = 39 Menit	28.	Pengamatan ke 28 = 38 Menit
14.	Pengamatan ke 14 = 39 Menit	29.	Pengamatan ke 29 = 40 Menit
15.	Pengamatan ke 15 = 40 Menit	30.	Pengamatan ke 30 = 39 Menit

Pada tabel diatas ditampilkan durasi waktu yang didapatkan dari pengamatan menggunakan fungsi Delay(1000).



Gambar 10. Hasil grafik data waktu fungsi Delay(1000)

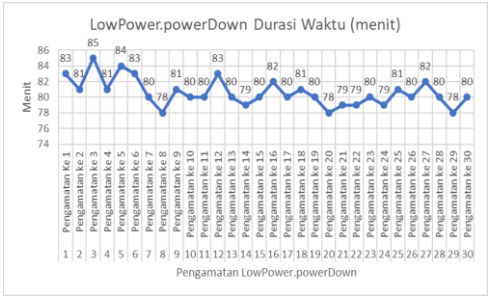
B. Pengujian Fungsi Low Power .PowerDown

Pada pengujian kali ini dimaksudkan untuk mengetahui apakah dengan fungsi yang digunakan yaitu fungsi Delay(1000) bisa melakukan efisiesi dengan baik atau tidak. Dengan diterapkan fungsi ini pada *Prototype* membuat *Prototype* efisien atau tidak dalam mengkonsumsi daya. Hasil dari pengujian ditampilkan dalam bentuk tabel dan grafik agar bisa dilihat secara jelas.

Tabel 2. Hasil pengamatan durasi menggunakan *LowPower.powerDown*

No	LowPower.powerDown		
1.	Pengamatan ke 1 = 83 Menit	16.	Pengamatan ke 16 = 82 Menit
2.	Pengamatan ke 2 = 81 Menit	17.	Pengamatan ke 17 = 80 Menit
3.	Pengamatan ke 3 = 85 Menit	18.	Pengamatan ke 18 = 81 Menit
4	Pengamatan ke 4 = 81 Menit	19.	Pengamatan ke 19 = 80 Menit
5.	Pengamatan ke 5 = 84 Menit	20.	Pengamatan ke 20 = 78 Menit
6.	Pengamatan ke 6 = 80 Menit	21.	Pengamatan ke 21 = 79 Menit
7.	Pengamatan ke 7 = 78 Menit	22.	Pengamatan ke 22 = 79 Menit
8.	Pengamatan ke 8 = 81 Menit	23.	Pengamatan ke 23 = 80 Menit
9.	Pengamatan ke 9 = 81 Menit	24.	Pengamatan ke 24 = 79 Menit
10.	Pengamatan ke 10 = 80 Menit	25.	Pengamatan ke 25 = 81 Menit
11.	Pengamatan ke 11 = 80 Menit	26.	Pengamatan ke 26 = 80 Menit
12.	Pengamatan ke 12 = 83 Menit	27.	Pengamatan ke 27 = 82 Menit
13.	Pengamatan ke 13 = 80 Menit	28.	Pengamatan ke 28 = 80 Menit

No	LowPower.powerDown		
14.	Pengamatan ke 14 = 79 Menit	29.	Pengamatan ke 29 = 78 Menit
15.	Pengamatan ke 15 = 80 Menit	30.	Pengamatan ke 30 = 80 Menit



Gambar 11. Hasil grafik data waktu menggunakan Fungsi LowPower.powerDown

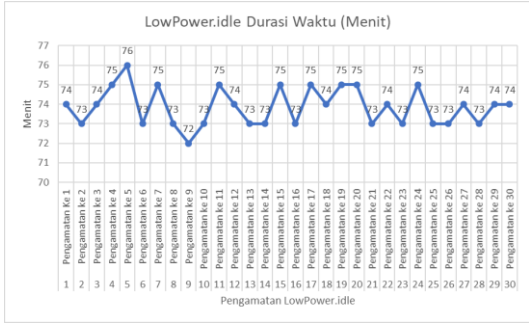
C. Pengujian Fungsi Low Power.Idle

Pada pengujian kali ini dimaksudkan untuk mengetahui apakah dengan fungsi yang digunakan yaitu fungsi LowPower.Idle bisa melakukan efisiesi dengan baik atau tidak. Dengan diterapkan fungsi ini pada *Prototype* membuat *Prototype* efisien atau tidak dalam mengkonsumsi daya. Hasil dari pengujian ditampilkan dalam bentuk tabel dan grafik agar bisa dilihat secara jelas.

Tabel 3. Tabel hasil pengamatan menggunakan Fungsi LowPower.idle

No	LowPower.idle		
1.	Pengamatan ke 1 = 72 Menit	16.	Pengamatan ke 16 = 75 menit
2.	Pengamatan ke 2 = 74 Menit	17.	Pengamatan ke 17 = 75 Menit
3.	Pengamatan ke 3 = 73 Menit	18.	Pengamatan ke 18 = 72 Menit
4	Pengamatan ke 4 = 70 Menit	19.	Pengamatan ke 19 = 74 Menit
5.	Pengamatan ke 5 = 73 Menit	20.	Pengamatan ke 20 = 75 Menit
6.	Pengamatan ke 6 = 73 Menit	21.	Pengamatan ke 21 = 73 Menit
7.	Pengamatan ke 7 = 75 Menit	22.	Pengamatan ke 22 = 74 Menit
8.	Pengamatan ke 8 = 76 Menit	23.	Pengamatan ke 23 = 71 Menit
9.	Pengamatan ke 9 = 72 Menit	24.	Pengamatan ke 24 = 75 Menit
10.	Pengamatan ke 10 = 70 Menit	25.	Pengamatan ke 25 = 76 Menit
11.	Pengamatan ke 11 = 74 Menit	26.	Pengamatan ke 26 = 74 Menit

No	LowPower.idle		
12.	Pengamatan ke 12 = 74 Menit	27.	Pengamatan ke 27 = 74 Menit
13.	Pengamatan ke 13 = 73 Menit	28.	Pengamatan ke 28 = 73 Menit
14.	Pengamatan ke 14 = 71 Menit	29.	Pengamatan ke 29 = 73 Menit
15.	Pengamatan ke 15 = 75 Menit	30.	Pengamatan ke 30 = 72 Menit



Gambar 12. Hasil grafik data waktu menggunakan Fungsi LowPower.idle

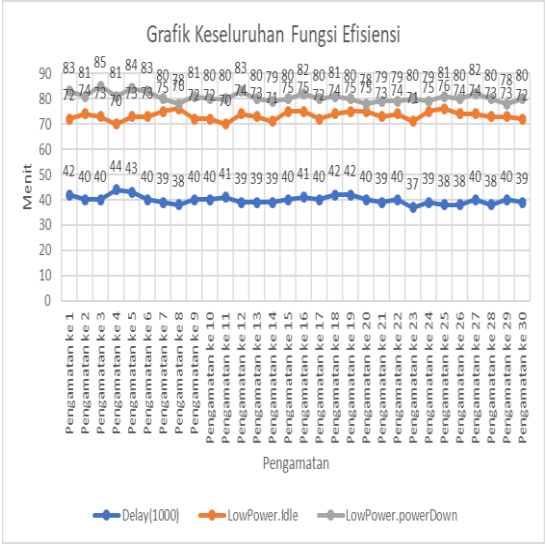
D. Nilai Ampere Dari Masing – Masing Fungsi

Dari percobaan dan pengamatan yang dilakakukan menghasilkan data nilai ampere yang muncul pada alat pengukuran Avometer, dari percobaan dan pengamatan tersebut digunakan alat pengukuran Avometer Analog, nilai yang dihasilkan relative hamper sama yaitu berkisar antara 90 – 101 mA, namun terjadi perbedaan kondisi jarum yang menunjukkan nilai pada Avometer, perbedaanya adalah ketika menggunakan Fungsi Delay(1000), maka Jarum stabil pada angka 101mA tidak ada penurunan nilai sama sekali. Namun ketika digunakan fungsi LowPower.idle, maka kondisi jarum mengalami penurunan nilai berkisar pada angka 94 mA, namun secara cepat kembali ke angka 101 mA kembali, kemudian pada percobaan menggunakan fungsi owPower.powerDown maka didapatkan hasil nilai yang juga sama 101 mA. Namun untuk kondisi jarumnya mengalami penurunan hingga angka berkisar 90 mA dengan gerakan yang lambat untuk kembali pada angka 101 mA, dan pada pengujian menggunakan Avometer digital untuk fungsi Delay(1000) menunjukkan nilai 42,3mA, dan menggunakan fungsi LowPower.idle menunjukkan nilai 95,2mA, sedangkan ketika menggunakan fungsi LowPower.powerDown menunjukkan nilai 92,1mA.

Terdapat perbedaan nilai terhadap Avometer digital dan analog, namun yang sangat terlihat signifikan adalah ketika menggunakan Avometer digital, perbandingan ini dimaksudkan untuk memberikan data dari perbedaan pengamatan daya ketika mengguakan Avometer digital dan analog. Untuk penyebabnya harus dikaji lebih ulang karena disini penulis hanya berfokus pada cara menghemat daya menggunakan algoritma efisiensi

Tabel 4. Hasil pengukuran Avometer

No	Fungsi	Avometer Analog	Avometer Digital
1.	Delay(1000)	101mA	42,3 mA
2.	LowPower.powerDown	94mA-101mA	95,2 mA
3.	LowPower.idle	90mA-101mA	92,1 mA



Gambar 13. Grafik keseluruhan fungsi

Saran

Dalam pengembangan selanjutnya dapat dilakukan integrasi dengan *smartphone* / android dalam ekstensi .apk sehingga lebih portable dan bisa dijangkau oleh setiap lapisan masyarakat dan pengamatan data bisa jadi lebih mudah. Dan ketika menguji *Prototype* ini sebaiknya jaringan dipastikan stabil, agar pembacaan dan pengiriman data di Database *Firebase* berjalan dengan lancar, serta dalam pengemasan *Prototype* sebaiknya menggunakan desain yang mengikuti bentuk *Prototype* sehingga alat terlihat lebih ringkas dan bagus

DAFTAR PUSTAKA

Musayyanah, Puspasari, I., & Susanto, P. 2018. Monitoring Target Heart Rate (THR) Untuk Latihan Lari Berbasis Internet Of Things. *Engineering and Sains Journal*, 87 - 94.

Martinez, B., Monton, M., Member, IEEE, & Prades, J. D. (2015). The Power Of Models: Modeling Power Consumption for IoT Device. *IEEE Sensors Journal*, 1-3.

Rintiasti, A., Bangsawan, H. T., Suhartono, A. A., Mahmudah, L., & Bahari, N. S. (2018). Pemantauan Proses Fermentasi Tembakau Menggunakan Platform IoT. *Desiminasi Hasil Litbang Baristand Industri Surabaya*, -.