

SKRIPSI

ANIMAL TRACKING BERBASIS IOT



POLITEKNIK NEGERI BALI

Oleh :

I Kadek Cahyadi Arta

NIM. 1815344020

**PROGRAM STUDI D4 TEKNIK OTOMASI
JURUSAN TEKNIK ELEKTRO
POLITEKNIK NEGERI BALI
2022**

LEMBAR PERSETUJUAN UJIAN SKRIPSI

ANIMAL TRACKING BERBASIS IOT

Oleh :

I Kadek Cahyadi Arta

NIM. 1815344020

Skripsi ini telah melalui Bimbingan dan Pengujian Hasil, disetujui untuk
diujikan pada Ujian Skripsi
di
Program Studi D4 Teknik Otomasi
Jurusan Teknik Elektro - Politeknik Negeri Bali

Bukit Jimbaran, 24/08/2022

Disetujui Oleh :

Dosen Pembimbing 1:

Dosen Pembimbing 2:



Ir. I Gusti Putu Mastawan Eka Putra, ST., MT.,
NIP. 197801112002121003



Ida Bagus Irawan Purnama, ST., M.Sc., Ph.D.,
NIP. 197602142002121001

LEMBAR PENGESAHAN SKRIPSI

ANIMAL TRACKING BERBASIS IOT

Oleh :

I Kadek Cahyadi Arta


NIM. 1815344020


Skripsi ini sudah melalui Ujian Skripsi pada tanggal 24/08/2022
dan sudah dilakukan Perbaikan untuk kemudian disahkan sebagai Skripsi
di
Program Studi D4 Teknik Otomasi
Jurusan Teknik Elektro - Politeknik Negeri Bali

Bukit Jimbaran, 27/09/2022


Disetujui Oleh :


Tim Penguji :


1. I Ketut Darminta, S.ST., MT.
NIP. 197112241994121001


2. Ir. Made Budiada, M.Pd.
NIP. 196506091992031002

Dosen Pembimbing :


1. Ir. Gusti Putu Masrawan Eka Putra, ST., MT.
NIP. 197801112002121003


2. Ida Bagus Irawan Purnama, ST., M.Sc., Ph.D.
NIP. 197602142002121003

Disahkan Oleh:

Jurusan Teknik Elektro




I. J. Wayan Raka Ardana, MT.
NIP. 196705021993031005

HALAMAN PERNYATAAN KEASLIAN KARYA SKRIPSI

Saya yang bertanda tangan di bawah ini, menyatakan bahwa Skripsi dengan judul:

Animal Tracking Berbasis IOT

adalah asli hasil karya saya sendiri.

Dengan ini saya menyatakan bahwa dalam naskah Skripsi ini tidak terdapat karya orang lain yang pernah diajukan untuk memperoleh gelar di suatu perguruan tinggi, dan atau sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah Skripsi ini, dan disebutkan dalam daftar pustaka.

Apabila saya melakukan hal tersebut di atas, dengan ini saya menyatakan menarik Skripsi yang saya ajukan sebagai hasil karya saya.

Bukit Jimbaran, 24/08/22

Yang menyatakan



I Kadek Cahyadi Arta

NIM.1815344020

ABSTRAK

Peternakan adalah suatu kegiatan pemeliharaan ataupun pengembangbiakkan hewan ternak guna mendapatkan manfaat dari hewan ternak tersebut. Pada peternakan ternak lepas, terdapat beberapa masalah yang cukup sulit dihadapi seperti ternak hilang, ternak mengalami kecelakaan, bahkan dapat terjerat hukuman pidana. Masalah-masalah ini muncul karena sulitnya kegiatan monitoring hewan ternak. Untuk mengatasi masalah-masalah tersebut diperlukan suatu sistem yang mampu melacak keberadaan dan keadaan hewan ternak secara real-time. Selain itu juga diperlukan suatu fitur untuk menjaga hewan ternak agar tetap berada dalam area peternakan. Karena area peternakan terletak jauh dari pemukiman maka diperlukan suatu media komunikasi yang mampu menjangkau area tersebut. Penelitian ini telah berhasil mengimplementasikan *IOT*, *GPS*, sensor *gyroscope* & *accelerometer* dan *LoRa* untuk melacak keberadaan dan keadaan. Pada pelacakan keberadaan hewan ternak dan didapat eror sebesar 0,055% pada latitude dan 0,011% pada longitude. Untuk pemindaian gerakan didapat tingkat keberhasilan sebesar 70% pada gerakan berbaring, 100% pada gerakan berdiri dan 80% untuk gerakan terjatuh. Dalam membangun pagar virtual digunakan konsep dasar bangun datar yang dipaduka dengan persamaan perpindahan dua buah vektor. Dalam penelitian ini didapat jarak eror sebesar 4,4 meter serta tingkat keberhasilan 90% pada area istirahat dan 80% pada area merumput. Pengaruh jarak terhadap kekuatan sinyal LoRa adalah berbanding terbalik dimana pada jarak 10 meter didapat nilai RSSI sebesar -64dBm, nilai ini terus mengecil hingga -131dBm pada jarak 100 meter.

Kata Kunci: Peternakan, IOT, GPS, LoRa, gyroscope & accelerometer

ABSTRACT

Livestock farming is an activity of raising or breeding livestock in order to benefit from it. In freelancing livestock farming, there are several problems that are quite difficult to deal with such as lost livestock, livestock experiencing accidents, and even being subject to criminal penalties. These problems arise because of the difficulty of monitoring livestock activities. To overcome these problems, a system is needed that is able to track the presence and condition of livestock in real-time. In addition, a feature is also needed to keep livestock in the farm area. Because the livestock area is located far from settlements, a communication medium is needed that is able to reach the area. This research has successfully implemented IoT, GPS, gyroscope & accelerometer sensors and LoRa to track whereabouts and circumstances. In tracking the presence of livestock and obtained an error of 0.055% at latitude and 0.011% at longitude. For motion scanning, the success rate is 70% for lying down, 100% for standing and 80% for falling. In building a virtual fence, the basic concept of a flat shape is used combined with the displacement equation of two vectors. In this study, the error distance was 4.4 meters and the success rate was 90% in the resting area and 80% in the grazing area. The effect of distance on LoRa signal strength is inversely proportional, at a distance of 10 meters the RSSI value is -64dBm, this value continues to decrease to -131dBm at a distance of 100 meters.

Keywords: *Livestock, IOT, GPS, LoRa, gyroscope & accelerometer*

KATA PENGANTAR

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena berkat rahmat dan hidayahNya penulis dapat menyelesaikan skripsi ini tepat pada waktunya. Skripsi ini disusun berdasarkan permasalahan yang penulis dapatkan untuk menjadikan sebuah penelitian yang dapat bermanfaat bagi yang membutuhkan, khususnya pada bidang Teknik Elektro, Program Studi D4 Teknik Otomasi.

Penyusunan skripsi ini merupakan salah satu hal yang wajib ditempuh dalam Program Studi D4 Teknik Otomasi. Dalam penyusunan skripsi ini penulis banyak mendapatkan bantuan dari berbagai pihak, oleh sebab itu penulis ingin mengucapkan terima kasih sebesar-besarnya kepada :

1. Bapak I Nyoman Abdi, SE., M.eCom., selaku Direktur Politeknik Negeri Bali.
2. Bapak Ir. Wayan Raka Ardana, MT., selaku Ketua Jurusan Teknik Elektro Politeknik Negeri Bali.
3. Bapak Ida Bagus Irawan Purnama, ST., M.Sc., Ph.D., selaku Ketua Program Studi D4 Teknik Otomasi Politeknik Negeri Bali, sekaligus Dosen Pembimbing yang telah memberikan bimbingan dan saran secara langsung selama penyusunan skripsi.
4. Bapak Ir. I Gusti Putu Mastawan Eka Putra, ST., MT., selaku Sekretaris Jurusan Teknik Elektro, sekaligus Dosen Pembimbing yang telah memberikan bimbingan dan saran secara langsung selama penyusunan skripsi.
5. Bapak dan Ibu Staf Jurusan Teknik Elektro Politeknik Negeri Bali yang telah banyak membantu dalam keperluan administrasi.
6. Keluarga yang selalu memberikan semangat dan dukungan selama penyusunan skripsi in
7. Rekan-rekan seperjuangan yang berstudi di Jurusan Teknik Elektro Politeknik Negeri yang telah membantu kelancaran dalam penyusunan skripsi ini.
8. Semua pihak yang telah membantu dalam proses penyusunan skripsi yang tidak dapat disebutkan satu per satu.

Dalam menyusun skripsi ini, penulis sangat menyadari banyaknya kekurangan yang terdapat pada skripsi ini. Oleh karena itu penulis sangat mengharapkan kritik dan saran yang membangun dari berbagai pihak, agar skripsi ini lebih baik lagi. Penulis berharap agar skripsi yang telah penulis susun dapat memberikan manfaat dan inspirasi bagi para pembaca maupun para penulis lainnya.

Tabanan, 20 Agustus 2022

Penulis

DAFTAR ISI

LEMBAR PERSETUJUAN UJIAN SKRIPSI.....	II
LEMBAR PENGESAHAN SKRIPSI.....	III
HALAMAN PERNYATAAN KEASLIAN KARYA SKRIPSI.....	IV
ABSTRAK.....	V
KATA PENGANTAR.....	VII
DAFTAR ISI.....	VIII
DAFTAR GAMBAR.....	X
DAFTAR TABEL.....	XII
DAFTAR LAMPIRAN.....	XIII
BAB I PENDAHULUAN.....	1
1.1. Latar Belakang.....	1
1.1. Perumusan Masalah.....	2
1.2. Batasan Masalah.....	3
1.3. Tujuan Penelitian.....	3
1.4. Manfaat Penelitian.....	4
BAB II TINJAUAN PUSTAKA.....	5
2.1. Penelitian Sebelumnya.....	5
2.2. Landasan Teori.....	6
2.2.1. Internet of things.....	6
2.2.2. NodeMCU ESP 8266.....	6
2.2.3. Arduino Nano.....	7
2.2.4. <i>LoRa</i>	7
2.2.5. <i>GPS</i>	8
2.2.6. Firebase.....	9
2.2.7. Kodular.....	9
2.2.8. <i>Geo-fencing</i>	10
2.2.9. Sensor gyroscope dan accelerometer.....	10
2.2.10. Panel Surya.....	11
BAB III METODE PENELITIAN.....	12
3.1. Movement Recognition Dari Pembacaan Sensor.....	12
3.2. Rancangan Sistem.....	15
3.2.1. Rancangan <i>Geo-fence</i>	16
3.2.2. Rancangan Perangkat <i>Client</i>	18
3.2.3. Rancangan Perangkat <i>Master</i>	19
3.3. Pembuatan Alat/Implementasi Sistem.....	20
3.3.1. Tahap Pembuatan <i>Hardware</i>	21
3.3.2. Tahap Pembuatan <i>Software</i>	21
3.4. Pengujian.....	25
3.4.2. Pengujian Kemampuan Melacak Lokasi.....	25
3.4.3. Pengujian Kemampuan Memindai Gerakan.....	26

3.4.4. Pengujian Keandalan <i>Geo-fencing</i>	26
3.4.5. Pengujian Pengaruh Jarak Terhadap Kekuatan Sinyal <i>LoRa</i>	26
BAB IV HASIL DAN PEMBAHASAN	27
4.1. Hasil	27
4.1.1. Alat <i>Animal Tracking</i> Berbasis <i>IOT</i>	27
4.1.1.1. <i>Hardware</i>	27
4.1.1.2. <i>Software</i>	30
4.1.2. Aplikasi <i>Animal Tracking</i> Berbasis <i>IOT</i>	32
4.1.2.1. Aplikasi	32
4.1.2.2. <i>Coding</i>	34
4.4.3. <i>Database</i>	35
4.2. Pengujian Sistem	37
4.2.1. Pengujian <i>Hardware</i>	37
4.2.2. Pengujian Program & Aplikasi	39
4.3. Pembahasan	40
4.3.1. Data Hasil Pengujian Kemampuan Lacak Lokasi	40
4.3.2. Data Hasil Pengujian Kemampuan Memindai Gerakan	41
4.3.3. Data Hasil Pengujian Keandalan <i>Geo-fencing</i>	43
4.3.4. Data Hasil Pengujian Pengaruh Jarak Terhadap Kekuatan Sinyal <i>LoRa</i>	44
BAB V PENUTUP	46
5.1. Kesimpulan	46
5.2. Saran	47
DAFTAR PUSTAKA	48

DAFTAR GAMBAR

Gambar 1. Iot ecosystem.....	6
Gambar 2. Pinout NodeMCU ESP8266.....	6
Gambar 3. Pinout arduino nano	7
Gambar 4. Sistem transmisi data LoRa.....	7
Gambar 5. Pinout Modul GPS neo-6m	8
Gambar 6. Tampilan firebase real-time database.....	9
Gambar 7. Tampilan kodular creator	9
Gambar 8. Geo-fencing.....	10
Gambar 9. Orientasi 6 degrees of freedom	10
Gambar 10. Panel surya 5V mini	11
Gambar 11. Flowchart penelitian.....	12
Gambar 12. Alur data sistem animal tracking berbasis Iot	15
Gambar 13. Blok diagram sistem animal tracking berbasis Iot	15
Gambar 14. Konsep membangun Geo-fence	16
Gambar 15. Bentuk lahan peternakan	16
Gambar 16. Bentuk lahan saat segitiga (x3,y3), (B), (x4,y4) diputar pada sumbu vertikal	17
Gambar 17. Wiring diagram perangkat client.....	18
Gambar 18. Skematik perangkat client	18
Gambar 19. Wiring diagram perangkat master	19
Gambar 20. Skematik perangkat master	19
Gambar 21. Flowchart perangkat client.....	22
Gambar 22. Flowchart perangkat master	23
Gambar 23. Flowchart aplikasi	24
Gambar 24. Komponen bagian kontrol perangkat client	27
Gambar 25. Komponen bagian battery charging perangkat client.....	28
Gambar 26. Komponen-komponen perangkat master.....	28
Gambar 27. Perangkat master animal tracking berbasis IOT.....	29
Gambar 28. Perangkat client animal tracking berbasis IOT	29
Gambar 29. Include library perangkat master dan client	30
Gambar 30. Void setup perangkat master dan client	31
Gambar 31. Coding timing pengiriman	31
Gambar 32. Coding pengiriman data ke firebase.....	32
Gambar 33. Coding eksekusi request.....	32
Gambar 34. Aplikasi animal tracking berbasis Iot.....	32
Gambar 35. Tampilan saat tombol LED ditekan sekali	33
Gambar 36. Reaksi dari perangkat client saat tombol LED ditekan sekali.....	33
Gambar 37. Pemberitahuan ternak keluar area peternakan.....	34
Gambar 38. Pemberitahuan ternak mengalami kecelakaan	34
Gambar 39. Frame screen initialize	34
Gambar 40. Frame firebase got value dan data changed.....	35
Gambar 41. Frame button LACAK dan button LED.....	35

Gambar 42. Firebase real-time database	36
Gambar 43. Airtable data storage.....	37
Gambar 44. Pengujian kerja LoRa	37
Gambar 45. pengujian GPS.....	38
Gambar 46. pengujian sensor MPU6050	38
Gambar 47. Serial monitor perangkat master saat pengujian kerja program	39
Gambar 48. Tampilan data pada aplikasi	39
Gambar 49. Grafik presentase error lacak lokasi	41
Gambar 50. Pengujian RSSI	45

DAFTAR TABEL

Tabel 1. Hasil pembacaan sensor MPU6050	13
Tabel 2. Data hasil pengujian kemampuan lacak lokasi	40
Tabel 3. Data hasil memindai gerakan Terjatuh ke depan, belakang dan kanan.....	41
Tabel 4. Data hasil memindai gerakan Terjatuh ke kiri, gerakan makan dan berdiri.....	42
Tabel 5. Presentase keberhasilan dan delay pengujian memindai gerakan.....	42
Tabel 6. Hasil memindai gerakan terjatuh	43
Tabel 7. Data hasil pengujian <i>geo-fencing</i>	43
Tabel 8. Data hasil pengujian pengaruh jarak terhadap kekuatan sinyal <i>LoRa</i>	44

DAFTAR LAMPIRAN

Lampiran 1. Tabel data hasil pengujian	50
Lampiran 2. Tabel pengenalan gerakan	51
Lampiran 3. Area peternakan	52
Lampiran 4. Tenak yang digunakan sebagai objek uji coba	52
Lampiran 5. Pengujian jarak eror geo-fancing.....	53
Lampiran 6. Coding Perangkat Client.....	54
Lampiran 7. Coding Perangkat Master.....	57

BABI PENDAHULUAN

1.1. Latar Belakang

Peternakan adalah suatu kegiatan pemeliharaan ataupun pengembangbiakkan hewan guna mendapatkan manfaat dari hewan ternak tersebut. Meskipun peternakan di Indonesia masih didominasi oleh ternak dengan kandang, namun tidak sedikit juga peternakan di Indonesia yang menggunakan sistem ternak lepas [1]. Sistem ternak lepas dilakukan dengan melepaskan hewan ternak pada suatu padang rumput atau lahan terbuka lainnya. Sistem ternak lepas masih sering dijumpai di Nusa Tenggara dan beberapa wilayah di Bali.

Sistem ternak seperti ini sangat menguntungkan peternak karena tidak perlu menyiapkan pakan untuk hewan ternaknya. Namun dibalik keuntungan tersebut, peternakan jenis ini juga sangat beresiko mulai dari kehilangan dan kecelakaan hewan ternak hingga pemantauan kesehatan yang tidak maksimal. Peternakan ternak lepas sangat sulit dipantau sehingga rawan kemalingan ataupun hewan hilang karena tersesat [1], [2]. Selain itu ternak juga rawan mengalami kecelakaan seperti terperosok ke jurang, tergelincir ataupun tersandung. Kecelakaan seperti itu memang tidak terlalu berbahaya apabila dapat ditangani dengan cepat. Namun pada peternakan ternak lepas biasanya hanya dikunjungi 2 kali sehari sehingga penanganan bila terjadi kecelakaan sering terlambat. Hal ini bisa menyebabkan cacat hingga kematian hewan ternak.

Menurut Revisi KUHP pasal 279, menyatakan bahwa: (1) Setiap orang yang membiarkan ternaknya berjalan di kebun, tanah perumputan, tanah yang ditaburi benih atau penanaman, atau tanah yang disiapkan untuk ditaburi benih atau ditanami dipidana dengan pidana denda paling banyak Kategori II. (2) Ternak sebagaimana dimaksud pada ayat (1) dapat dirampas oleh negara. Bercermin dari RKUHP tersebut semua peternak di Indonesia harus lebih berhati-hati agar hewan ternaknya tetap aman dan tidak sampai merusak lahan orang lain hingga menimbulkan kerugian.

Untuk mengurangi risiko kehilangan hewan ternak, diperlukan suatu sistem yang dapat memonitoring posisi hewan ternak secara nirkabel dan *real-time*. Agar hewan ternak yang mengalami kecelakaan dapat ditangani dengan cepat, diperlukan sebuah sistem yang mampu menganalisa gerakan hewan ternak. Apabila terindikasi kecelakaan maka sistem akan mengirimkan peringatan dini sehingga bisa langsung ditangani. Untuk menghindari jeratan RKUHP pasal 279 dan menjaga agar hewan ternak tetap berada di dalam area peternakan,

diperlukan sistem pagar virtual. Pagar virtual bisa dibangun dengan sistem *geo-fencing* yang akan memberi peringatan dini apabila hewan tersebut melewati batas–batas *geo-fencing*[3]. Selain ketiga hal di atas, sistem ini juga harus mampu bekerja di area yang sulit sinyal seluler karena kebanyakan peternakan terletak jauh dari pemukiman.

Untuk mewujudkan sistem seperti yang disebutkan di atas, salah satu caranya adalah dengan mengimplementasikan *Internet of things (Iot)*, *movement recognition* dan *geo-fencing* dengan *Global Positioning Sistem (GPS)* yang terintegrasi dengan teknologi *Long Range (LoRa)*. Dengan implementasi *Iot*, diharapkan pemantauan hewan ternak dapat dilakukan secara *real-time* dan dapat dipantau dari jarak yang jauh[1], [4]. Dengan penggunaan sistem *movement recognition*, keadaan hewan ternak diharapkan dapat terpantau sehingga penanganan apabila terjadi kecelakaan dapat dipercepat. Penggunaan *GPS* diharapkan mampu memberi informasi lokasi secara akurat[5], [6], [7]. Selain itu penggunaan *GPS* juga diharapkan mampu membangun sistem *geo-fencing* sehingga hewan ternak tetap berada dalam area peternakan[3][6]. Dengan imlementasi *LoRa*, diharapkan sistem ini dapat menjangkau area yang sulit sinyal seluler[5], [8], [9], [10]. Untuk media *User Interface (UI)* akan dibuat berupa aplikasi *Smartphone* untuk memudahkan pengguna.

Penelitian ini akan membuat suatu sistem *Animal Tracking* berbasis *Iot*. Sistem ini diharapkan mampu memonitoring keberadaan hewan ternak. Selain keberadaan, sistem ini juga diharapkan mampu memonitoring keadaan hewan ternak dan memberi peringatan dini apabila hewan ternak mengalami kecelakaan. Selain itu sistem ini juga diharapkan mampu memberi peringatan dini kepada peternak apabila hewan ternak keluar dari area peternakan. Alat yang dibuat adalah beberapa *client* berupa kalung hewan, sebuah perangkat *master/gateway* serta media *UI* berupa aplikasi *Smartphone* yang dibuat dengan Kodular. Penelitian ini menggunakan objek penelitian berupa ternak sapi lepas di area desa Kerambitan, Tabanan, Bali.

1.2. Perumusan Masalah

Berdasarkan latar belakang di atas, maka dapat dirumuskan permasalahan dalam penelitian ini yaitu:

- a. Bagaimanakah implementasi sistem *Iot* untuk memonitoring keberadaan hewan ternak secara nirkabel?

- b. Bagaimanakah implementasi *movement recognition* untuk menganalisa gerakan hewan ternak?
- c. Bagaimanakah implementasi modul *GPS* dalam membangun pagar virtual dan menganalisis pergerakan temporal hewan ternak?
- d. Bagaimanakah pengaruh jarak terhadap kekuatan sinyal komunikasi modul *LoRa* SX1276 dalam memonitoring keberadaan hewan ternak?

1.3. Batasan Masalah

Dalam penelitian ini, ruang lingkup penelitian akan dibatasi pada:

- a. Perangkat *LoRa* yang digunakan adalah RA-01H SX1276 dengan frekuensi 913 MHz dengan transmisi topologi star.
- b. Perangkat *GPS* yang digunakan adalah modul *GPS* NEO-6M, sedangkan perangkat sensor *gyroscope* dan *accelerometer* yang digunakan adalah modul sensor MPU6050.
- c. Penelitian menggunakan objek 2 ekor sapi yang dilakukan di desa Kerambitan, Tabanan, Bali.
- d. Area *geo-fencing* yang digunakan mengikuti bentuk area peternakan.
- e. Pengujian akurasi lokasi menggunakan Google Maps sebagai acuan.
- f. Gerakan hewan ternak yang diuji adalah berdiri, makan dan terjatuh.
- g. Pengujian hewan terjatuh dilakukan dengan simulasi, sedangkan pengujian pengaruh jarak berfokus pada pengaruh jarak serta halangan terhadap nilai *RSSI*

1.4. Tujuan Penelitian

Berdasarkan uraian latar belakang dan rumusan masalah di atas, maka tujuan dari penelitian ini yaitu:

- a. Dapat membuat sistem monitoring keberadaan hewan ternak secara nirkabel.
- b. Dapat membuat sistem *movement recognition* untuk mengetahui keadaan dari hewan ternak.
- c. Dapat membuat sistem pagar virtual yang mampu memberikan informasi pergerakan temporal hewan ternak di dalam area peternakan.

- d. Dapat mengetahui perbandingan jarak transfer data dengan kekuatan sinyal *LoRa* SX1276 untuk mengetahui efektivitas kerja *LoRa* dalam area peternakan.

1.5. Manfaat Penelitian

Berikut manfaat dari dilaksanakannya penelitian ini adalah:

- a. Membantu peternak dalam mengamankan hewan ternak dari hal-hal yang tidak diinginkan.
- b. Mengetahui karakteristik area untuk pengaplikasian teknologi *Iot* dalam bidang peternakan.

1.6. Sistematika Penulisan

Penelitian skripsi ini terdiri dari 5 bab, yaitu:

- a. Bab I Pendahuluan
Menguraikan tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.
- b. Bab II tinjauan Pustaka
Menguraikan tentang penelitian sebelumnya dan landasan teori yang berisi definisi dari komponen komponen yang digunakan dapa alat yang dibangun.
- c. Bab III Metode Penelitian
Menguraikan perancangan sistem, pembuatan alat dan prosedur pengujian alat.
- d. Bab IV Hasil dan Pembahasan
Menguraikan dan menganalisis data yang didapat dari pengujian alat.
- e. Bab V Penutup
Menguraikan kesimpulan dari penelitian, serta saran-saran yang diperlukan untuk penelitian selanjutnya.

BAB II

TINJAUAN PUSTAKA

2.1. Penelitian Sebelumnya

Beberapa penelitian yang relevan dengan *Animal Tracking*, *geo-fencing* dan *LoRa* yang sudah dilakukan antara lain:

Penelitian sebelumnya [1] membuat sistem pelacak lokasi hewan ternak dengan sistem komunikasi *LoRa*. Penelitian ini menggunakan Arduino Uno, modul *GPS NEO-6M* dan *Dragoino LoRa Shield 915Mhz* sebagai Node. Untuk *Gateway* menggunakan Arduino Uno, *Dragoino LoRa Shield 915Mhz*, *Ground Plane Antenna FPV Telemetry* dan *SIM900 mini*. Penelitian ini hanya berfokus pada akurasi lokasi dan pengaruh jarak terhadap kekuatan sinyal. Perlu adanya suatu fitur yang mampu memberi notifikasi pada peternak jika ada hal yang tidak diinginkan terjadi pada hewan ternak.

Penelitian [3] meneliti implementasi *Geo-fencing* untuk mengawasi kendaraan pengirim paket ekspedisi. Penelitian ini dilakukan dengan membuat sebuah aplikasi mobile berbasis Android dengan memanfaatkan Google API. Penelitian ini berfokus monitoring di jalur utama pengiriman dengan ruang lingkup yang terlalu luas sehingga kurang spesifik dalam menentukan lokasi aktual dari kendaraan pengirim paket. Selain itu, penelitian ini juga tidak memberi solusi seandainya perangkat *smartphone* yang digunakan kehabisan daya baterai.

Penelitian [4] membuat sistem pelacak lokasi kendaraan berbasis *GPS* yang menggunakan *SMS Gateway* untuk berkomunikasi dengan *smartphone*. Sistem ini menggunakan mikrokontroler Arduino Nano, dan *SIM800L* sebagai media komunikasi. Penelitian ini berfokus pada pelacakan lokasi secara otomatis, namun penanganannya masih secara manual yaitu melalui *SMS*.

Penelitian [8] membuat sistem *emergency button* yang dapat membantu para pendaki jika terjadi hal yang tidak diinginkan. Sistem dirancang dengan membuat sebuah node dan sebuah *gateway*, yang keduanya sama-sama terdiri dari Arduino Uno sebagai mikrokontroler dan *LoRa RFM9x* sebagai media komunikasi. Sistem ini hanya akan memberi peringatan ke pos pendakian apabila ada yang memencet tombol *emergency* pada node. Peringatan yang diberikan hanya berupa suara *buzzer* dan tanpa informasi lebih lanjut. Alangkah baiknya apabila pada sistem ini diimplementasikan sistem *GPS* agar tim penyelamat bisa setidaknya mendapat informasi titik terakhir yang dilalui pendaki.

Penelitian ini diharapkan mampu mengimplementasikan *LoRa* dan *GPS* dengan baik yang dipadukan dengan *User Interface* yang dapat memudahkan *user*. Penelitian ini akan mengoptimalkan penggunaan *LoRa* pada lahan peternakan ternak lepas. Perbedaan penelitian ini terletak pada perangkat dan sistem interface yang digunakan. Perangkat yang digunakan antara lain: NodeMCU ESP8266 dan Arduino Nano sebagai mikrokontroler, Modul *LoRa* SX1276 sebagai media komunikasi antara node dengan *gateway*, modul *GPS* NEO-6M sebagai pelacak lokasi serta aplikasi Kodular sebagai media *interface*.

2.2. Landasan Teori

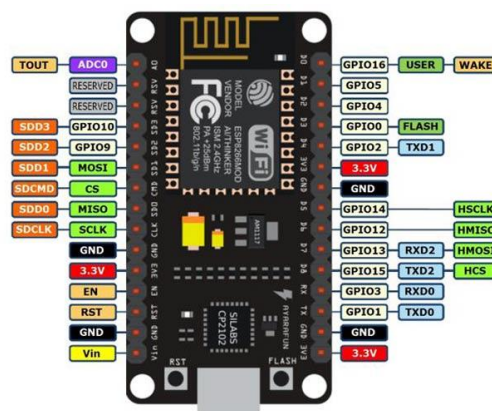
2.2.1. Internet of things



Gambar 1. Iot ecosystem

Internet of things merupakan suatu komunikasi antar perangkat elektronik dengan perangkat lain melalui jaringan internet [4]. Komunikasi ini memungkinkan terjadinya komunikasi antara sensor, perangkat elektrik atau mekanik dan pengguna meski terpisah dalam jarak yang jauh.

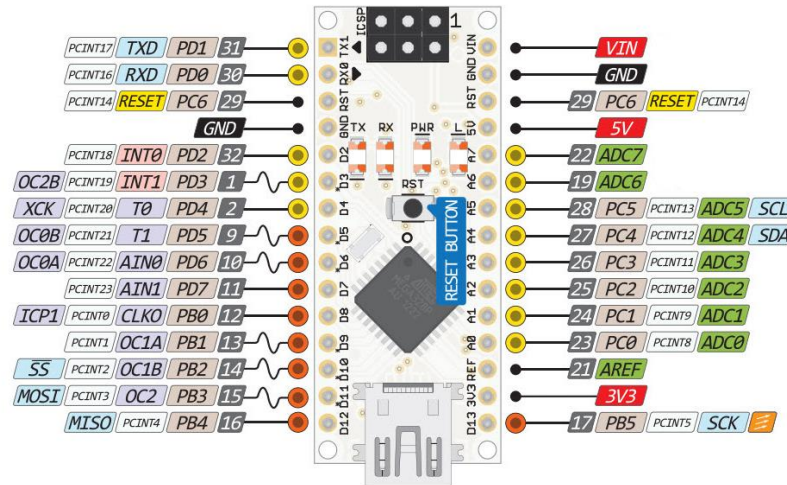
2.2.2. NodeMCU ESP 8266



Gambar 2. Pinout NodeMCU ESP8266

NodeMCU ESP8266 merupakan sebuah *open source* platform *IoT* dan pengembangan kit yang menggunakan bahasa pemrograman Lua untuk membantu dalam membuat prototipe produk *IoT* atau bisa dengan memakai sketch dengan arduino IDE. NodeMCU berukuran panjang 2.83cm, lebar 2.54cm, dan berat 7 gram. *Board* ini sudah dilengkapi dengan fitur *WiFi* dan *Firmware*-nya yang bersifat *open source*.

2.2.3. Arduino Nano

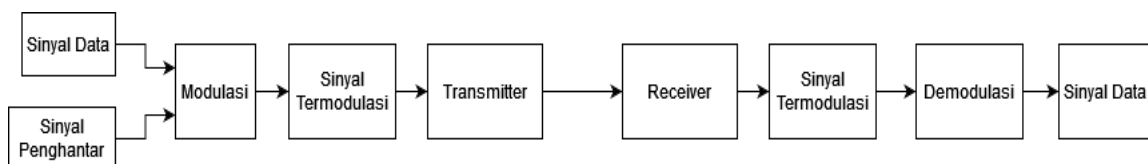


Gambar 3. Pinout arduino nano

Arduino nano merupakan sebuah perangkat elektronik yang bersifat open-source. Arduino nano menggunakan ATmega328 sama halnya dengan Arduino uno [4], [11]. Sesuai dengan namanya, Arduino nano memiliki ukuran yang jauh lebih kecil dari Arduino uno yaitu panjang 45mm dan lebar 18mm. Ukuran yang sangat kecil membuat Arduino nano menjadi pilihan yang tepat untuk project yang memerlukan perangkat berukuran kecil.

2.2.4. LoRa

LoRa merupakan suatu produk sarana komunikasi jarak jauh yang dibuat dan dipatenkan oleh Semtech Corporation [10], [12]. *LoRa* termasuk dalam sistem komunikasi *Low Power Wide Area Network (LPWAN)* yang memerlukan sedikit energi namun mampu melakukan transmisi jarak jauh [13]. *LoRa* mentransmisikan data menggunakan sinyalnya sendiri. Gambar 2 merupakan sistem transmisi data *LoRa*.

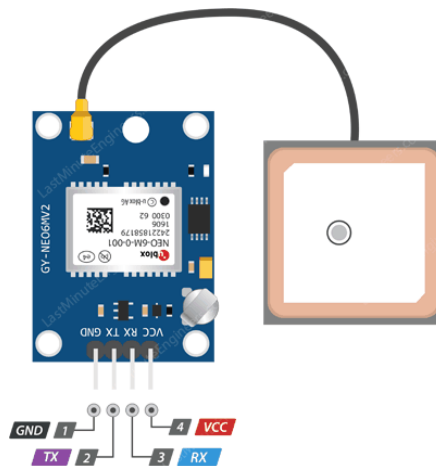


Gambar 4. Sistem transmisi data LoRa

Sinyal data yang akan ditransmisikan akan dimodulasikan dengan sinyal penghantar. Sinyal yang sudah termodulasi kemudian dikirim oleh perangkat *transmitter* dan diterima oleh *receiver*. Sinyal yang diterima adalah sinyal yang termodulasi. Sinyal termodulasi ini perlu didemodulasi untuk mendapatkan kembali sinyal data.

LoRa beroperasi pada frekuensi berbeda untuk tiap benua. *LoRa* benua Asia menggunakan frekuensi 433MHz, namun Kominfo menetapkan standar frekuensi *LoRa* di Indonesia dengan frekuensi 913MHz. Untuk Eropa menggunakan frekuensi 868MHz sedangkan untuk Amerika Utara menggunakan 915MHz [9]. *LoRa* SX1276 Ra-02 merupakan satu dari beberapa produk *LoRa transceiver* buatan Semtech Corporation. *LoRa transceiver* jenis ini mampu melakukan transmisi data hingga jarak belasan kilometer dengan konsumsi daya yang tergolong rendah. SX1276 Ra-2 bekerja pada frekuensi 137 – 960MHz sehingga cocok digunakan di Indonesia. Selain SX1276, Semtech Corporation juga memproduksi varian lain seperti: SX1272 dan SX1273 yang bekerja pada frekuensi 860 – 1000MHz, SX1278 yang bekerja pada frekuensi 137 – 586MHz dan SX1277 yang bekerja pada frekuensi 137 – 1020MHz.

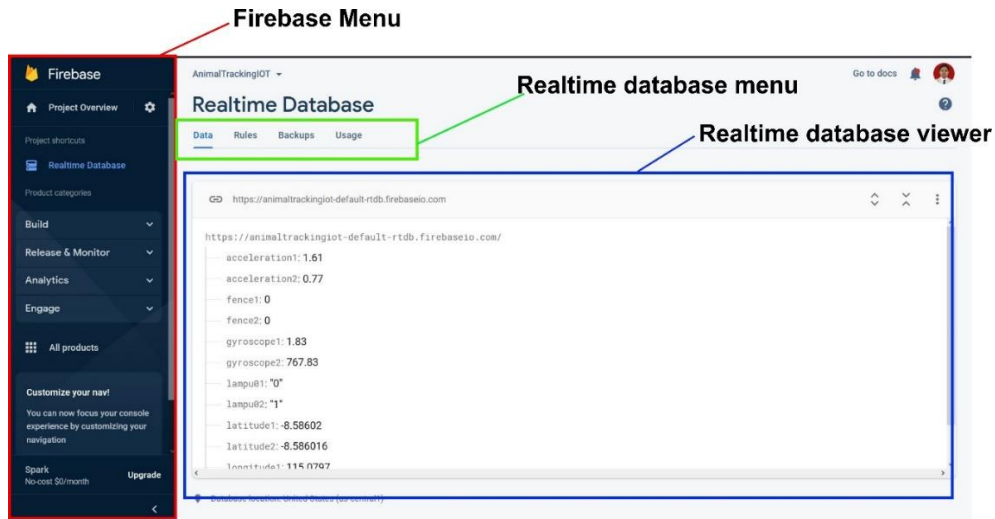
2.2.5. GPS



Gambar 5. Pinout Modul GPS neo-6m

GPS adalah suatu teknologi yang mampu menentukan lokasi objek pada permukaan bumi. *GPS* dapat bekerja dengan memanfaatkan sinyal dari setidaknya 3 satelit untuk menentukan posisi lintang dan bujur suatu objek [6], [7]. *GPS* juga bisa menggunakan setidaknya 4 satelit untuk dapat menentukan ketinggian objek [14]. Modul *GPS* Neo-6M merupakan perangkat *transceiver GPS* yang paling sering digunakan. Perangkat ini tergolong *low power* karena bisa beroperasi pada tegangan 3.3V. Selain itu, perangkat ini juga sangat mudah dioperasikan karena sudah ada *library* *TinyGPS++* pada Arduino IDE.

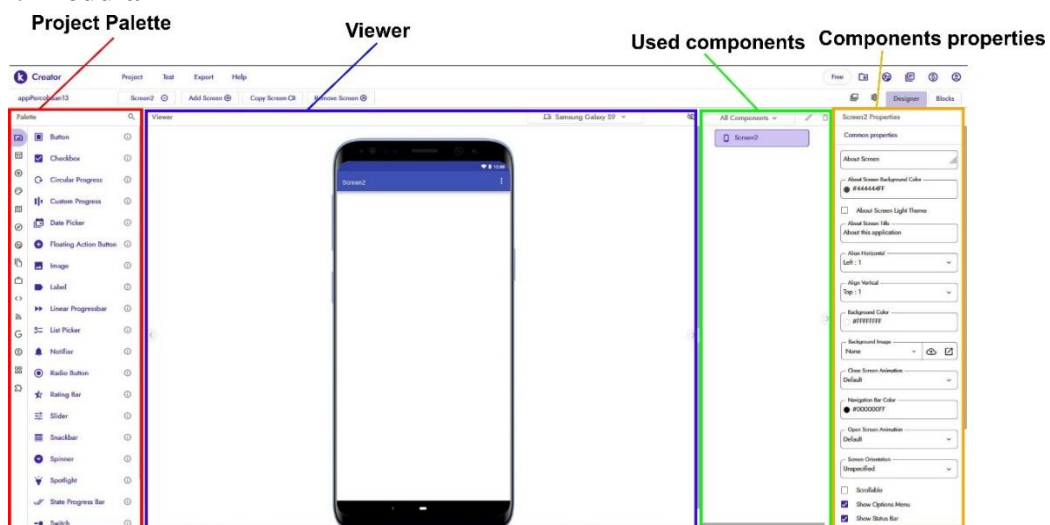
2.2.6. Firebase



Gambar 6. Tampilan firebase real-time database

Firebase merupakan suatu produk Google yang dibuat untuk membantu para pengembang dalam mengembangkan sebuah aplikasi. Firebase pertama kali dibuat pada tahun 2011 oleh Andrew Lee dan James Tramppin, yang kemudian diakuisisi oleh Google pada tahun 2014. Firebase dibuat dengan tujuan memberikan layanan kepada para pengembang untuk fokus mengembangkan aplikasi tanpa perlu memberikan *effort* yang besar. Karenanya Kodular termasuk ke dalam layanan *Backend as a Service (BaaS)*. *BaaS* merupakan layanan *Cloud computing* yang di sediakan oleh sebuah perusahaan untuk memudahkan pengembang untuk megembangkan aplikasi mobile. Firebase menawarkan *real-time database* yang mampu menyimpan data dan sinkronisasi ke banyak pengguna.

2.2.7. Kodular



Gambar 7. Tampilan kodular creator

Kodular merupakan suatu situs yang menyediakan *tools* untuk membuat suatu aplikasi android. Kodular menggunakan blok–blok programing yang didesign seperti *puzzle*. Hal ini sangat memudahkan pengguna dalam membangun program sesuai keinginan. Meski sepiantas terlihat sama seperti MIT App Inventor, namun Kodular sedikit lebih unggul karena fitur Kodular *extension* dan Kodular *store*. Kedua fitur ini sangat memudahkan pengembang untuk membuat dan mengunggah aplikasi yang sudah dibuat.

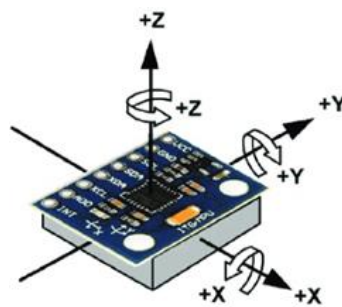
2.2.8. Geo-fencing



Gambar 8. *Geo-fencing*

Geo-fencing merupakan suatu sistem yang memanfaatkan salah satu fitur *GPS*. *Geo-fencing* dibuat dengan membuat beberapa titik pada peta yang nantinya akan menjadi batas – batas virtual pada peta [3]. Apabila suatu objek yang dipantau bergerak melewati batas – batas tersebut maka sistem akan mengidentifikasi bahwa objek tersebut sudah keluar area.

2.2.9. Sensor gyroscope dan accelerometer

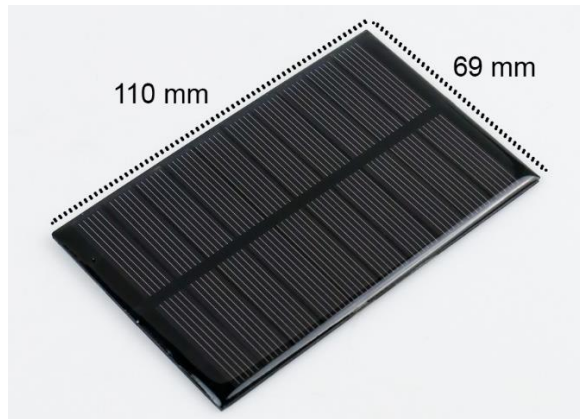


Gambar 9. *Orientasi 6 degrees of freedom*

Sensor *gyroscope* dan *accelerometer* adalah sebuah sensor yang mampu mendeteksi tiga poros *gyro* dan tiga poros *acceleration*. tiga poros *gyro* berupa nilai *x*, *y*, *z* yang diterjemahkan menjadi *roll*, *pitch* dan *yaw* [15]. *Roll*, *pitch* dan *yaw* digunakan untuk menentukan orientasi dari benda yang diamati. tiga poros *acceleration* juga berupa nilai *x*, *y*, *z*, namun pada *accelerometer* nilai ini diterjemahkan menjadi arah serta percepatannya.

Sensor MPU6050 merupakan salah satu sensor yang mampu mendeteksi tiga poros baik itu *gyro* ataupun *acceleration*. Hal itu membuat sensor ini mampu mendeteksi 6 *degrees of freedom* yaitu posisi x, y dan z serta orientasi *roll*, *pitch* dan *yaw*.

2.2.10. Panel Surya

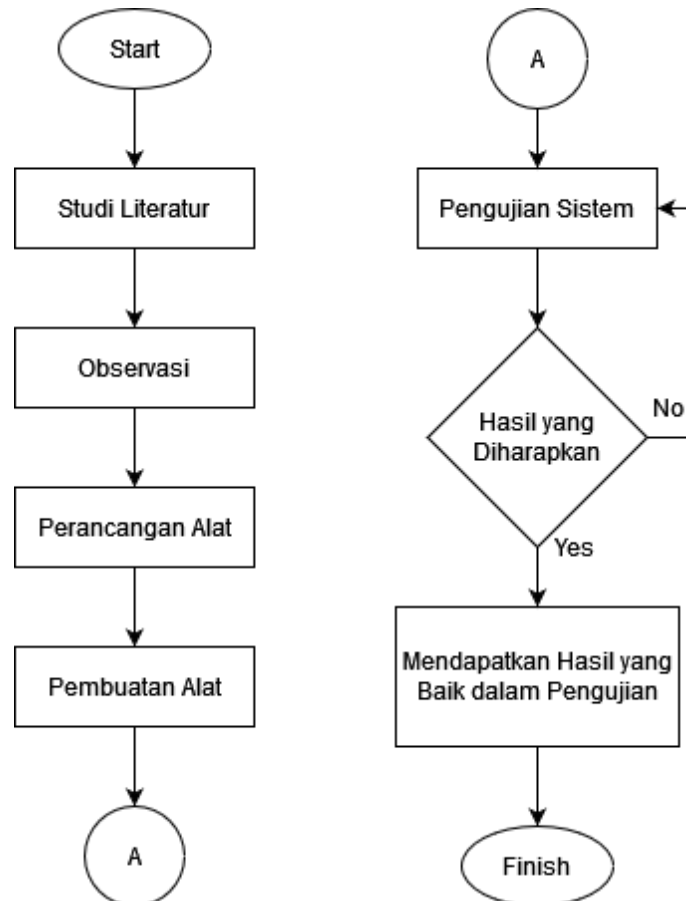


Gambar 10. Panel surya 5V mini

Panel surya merupakan suatu perangkat yang mampu mengubah energy sinar matahari menjadi energi listrik. Panel surya terdiri dari beberapa perangkat yang lebih kecil. Perangkat penyusun panel surya ini disebut sel surya [16], [17]. Sel surya memanfaatkan efek fotovoltaic untuk mengubah energi sinar matahari menjadi tegangan listrik. Sebuah sel surya mampu menghasilkan tegangan antara 0,45V hingga 0,6V [18], [19], [20]. Untuk mendapat panel surya dengan tegangan 5V, para produsen biasanya menggabungkan 10 sel surya yang dirangkai secara seri.

BAB III METODE PENELITIAN

Metode penelitian diawali dengan studi literatur guna mengumpulkan informasi yang dapat membantu penelitian ini. Selanjutnya dilakukan metode pengumpulan data berupa metode observasi guna mengamati dan mencatat secara sistematis terhadap gejala yang tampak pada objek penelitian. Kemudian dilakukan analisa kebutuhan sistem, baik itu perangkat keras ataupun perangkat lunak. Selanjutnya dilakukan perancangan sistem sesuai dengan analisa kebutuhan sistem. Setelah itu dilanjutkan dengan proses integrasi dan implementasi sistem. Selanjutnya dilakukan pengujian komponen-komponen yang digunakan hingga pengujian sistem secara keseluruhan. Gambar 11 menunjukkan *flowchart* dari penelitian ini.



Gambar 11. Flowchart penelitian

3.1. Movement Recognition Dari Pembacaan Sensor

Pada tahap observasi akan dilakukan pengumpulan data gerakan sapi untuk membuat dataset sistem *movement recognition*. Pengumpulan data meliputi gerakan jatuh, berdiri dan makan. Untuk gerakan terjatuh dilakukan dengan simulasi, sedangkan untuk gerakan berdiri

dan makan dilakukan langsung pada hewan ternak. Pengumpulan data gerakan terjatuh akan dilakukan dengan menyimulasikan gerakan terjatuh ke arah depan, kanan, kiri dan belakang masing-masing sebanyak 10 kali pada sebuah boneka berbentuk sapi yang sudah dipasangi kalung dengan sensor MPU6050.

Data keluaran sensor dari tiap-tiap percobaan akan dicatat. Nilai terendah dan tertingginya akan digunakan sebagai dataset untuk mengenali gerakan terjatuh. Pengumpulan data gerakan berdiri dan makan kurang lebih sama seperti gerakan terjatuh, hanya saja objek yang digunakan adalah hewan ternak secara langsung. Berikut adalah Tabel 1 yang merupakan tabel hasil pembacaan sensor MPU6050.

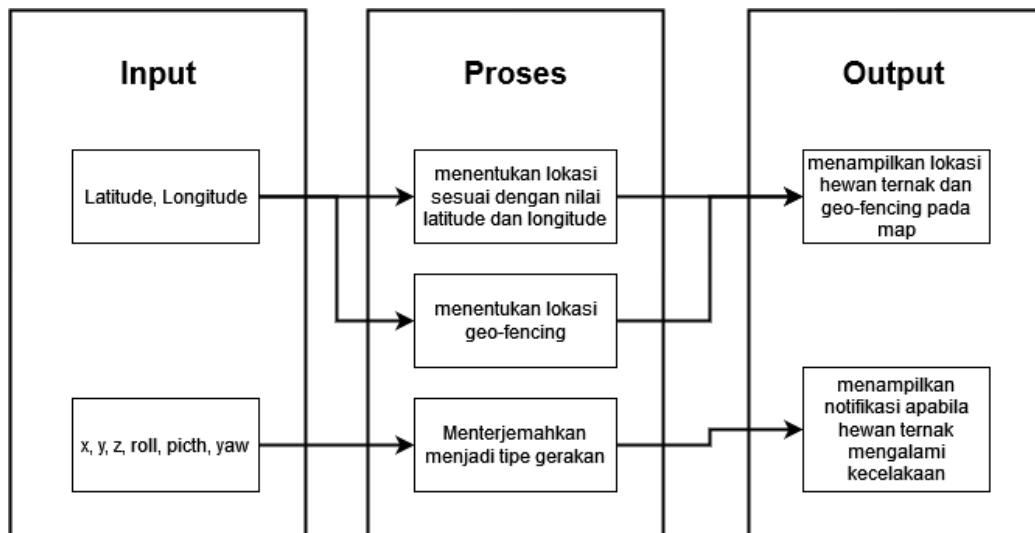
Tabel 1. Hasil pembacaan sensor MPU6050

A. Terjatuh Ke Depan						
Percobaan ke-	X	Y	Z	Roll	Pitch	Yaw
1	-0.27246	0.09227	1.34186	14.65004	22.11966	-9.86089
2	-0.2871	0.00072	1.27253	3.21493	60.80669	5.11621
3	0.08716	-0.11305	-0.38616	12.40577	27.61585	-12.3342
4	-0.24633	-0.01735	1.67536	6.02409	117.3716	4.90247
5	-0.01269	-0.01954	0.96516	3.61187	4.63875	-6.01356
6	-0.27246	0.09227	1.34186	14.65004	22.11966	-9.86089
7	-0.01269	-0.01954	0.96516	3.61187	30.92497	179.8643
8	-0.19213	0.00438	0.65119	32.72638	-10.7048	4.04751
9	0.08716	-0.11305	-0.38616	12.40577	27.61585	-12.3342
10	-0.2871	0.00072	1.27253	3.21493	60.80669	5.11621
Nilai terendah	-0.2871	-0.11305	-0.38616	3.21493	-10.7048	-12.3342
Nilai tertinggi	0.08716	0.09227	1.67536	32.72638	117.3716	179.8643
B. Terjatuh Ke Belakang						
1	-2.35411	-0.27707	2.9527	-133.16	336.7489	-91.6318
2	-1.85533	-0.3215	1.62408	29.86278	255.2069	-55.4028
3	-0.08067	-0.13937	0.1087	-73.6639	255.2069	-55.4028
4	-2.35411	-0.27707	2.9527	-133.16	336.7489	-91.6318
5	-0.213	-0.28415	0.30475	-39.893	417.0237	-70.5783
6	-0.08067	-0.13937	0.1087	-73.6639	-307.16	50.12397
7	-0.68345	-0.07101	0.83331	29.16049	-307.16	50.12397
8	-1.85533	-0.3215	1.62408	29.86278	255.2069	-55.4028
9	-0.76402	-0.03732	2.53937	57.81697	91.54278	0.27664
10	-0.68345	-0.07101	0.83331	29.16049	-307.16	50.12397
Nilai terendah	-2.35411	-0.3215	0.1087	-133.16	-307.16	-91.6318
Nilai tertinggi	-0.08067	-0.03732	2.9527	57.81697	417.0237	50.12397
C. Terjatuh Ke Kanan						
1	-0.24309	0.23884	0.88612	13.17301	-68.7993	99.66259
2	-0.76481	-0.4909	1.08046	46.63866	28.89536	-118.566
3	-0.88298	-0.4328	0.57509	-30.4606	56.26941	-204.597
4	-0.09196	0.3343	0.26479	71.53942	-102.586	365.0519

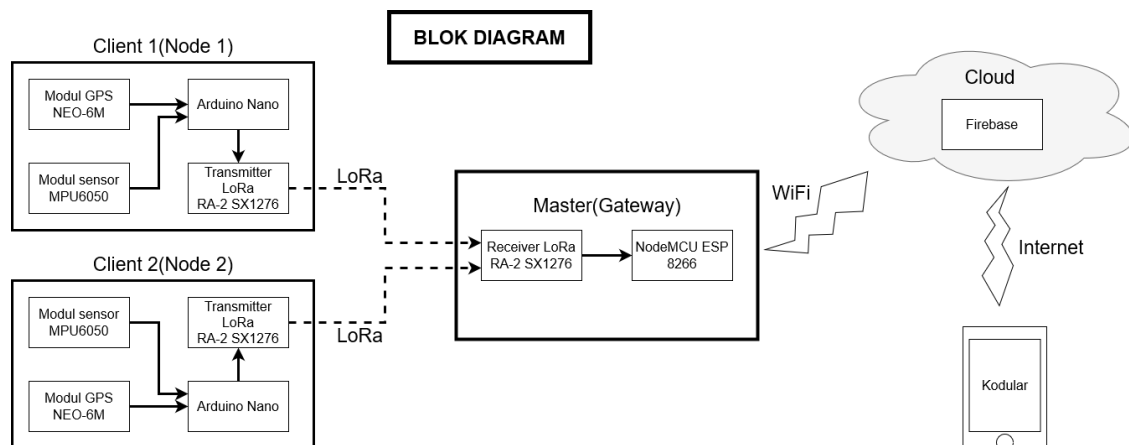
5	-0.88298	-0.4328	0.57509	-30.4606	56.26941	-204.597
6	-1.30705	-0.3178	0.91884	-76.6438	-43.0894	259.6015
7	-1.50285	-0.87444	0.26479	71.53942	-102.586	-229.253
8	-0.09196	0.3343	0.80238	23.49362	-24.7382	4.6397
9	-1.30705	-0.3178	0.91884	-76.6438	-43.0894	-118.566
10	-0.99943	-0.94647	0.28236	0.7608	-68.7993	99.66259
Nilai terendah	-1.50285	-0.94647	0.26479	-76.6438	-102.586	-229.253
Nilai tertinggi	-0.09196	0.3343	1.08046	71.53942	56.26941	365.0519
D. Terjatuh Ke kiri						
1	-0.99812	0.89296	0.91595	104.9939	60.7723	-192.044
2	-0.91487	1.30019	0.84246	12.23049	-7.57884	50.4292
3	-0.96663	1.33339	0.76214	15.43659	-27.9605	79.74217
4	-0.99812	0.89296	0.91595	104.9939	60.7723	-192.044
5	-1.17488	1.89443	0.98797	88.10835	-1.62464	-16.5784
6	-0.4659	0.89296	0.91595	104.9939	60.7723	-192.044
7	-0.4471	0.05825	0.99285	-17.7695	1.44406	166.017
8	-0.99812	0.89296	0.91595	104.9939	60.7723	-192.044
9	-0.96663	1.30019	0.84246	12.23049	-7.57884	50.4292
10	-0.34724	0.9579	1.0263	-14.4565	-4.49487	111.1315
Nilai terendah	-1.17488	0.05825	0.76214	-17.7695	-27.9605	-192.044
Nilai tertinggi	-0.34724	1.89443	1.0263	104.9939	60.7723	166.017
E. Gerakan Berbaring						
1	-0.10069	0.0837	0.93982	-12.9878	24.81829	-12.7855
2	0.09462	-0.40092	0.77014	-4.62137	-22.1435	15.38244
3	0.10205	0.01234	1.02693	-26.3859	-2.1218	-21.4979
4	-0.01392	-0.16222	1.02742	68.99695	-19.8382	43.79466
5	0.00551	0.00313	0.99646	0.5542	-1.68553	2.25111
6	0.10205	0.01234	1.02693	-26.3859	-2.1218	-21.4979
7	0.00551	0.00313	0.99646	18.32828	83.5047	2.25111
8	0.00038	0.04659	1.00379	-11.1863	33.29157	52.17634
9	0.12695	0.12269	1.05257	-27.6836	12.30568	-50.2307
10	0.28222	0.35926	1.13142	-109.607	1.11484	-17.9712
Nilai terendah	-0.10069	-0.40092	0.77014	-109.607	-22.1435	-50.2307
Nilai tertinggi	0.28222	0.35926	1.13142	68.99695	83.5047	52.17634
F. Gerakan Berdiri						
1	-1.62165	-0.24768	0.49728	-2.0534	-9.10598	13.31982
2	-1.67292	-0.20423	0.39498	-27.9465	-5.88461	13.51829
3	-0.64436	-0.18909	0.35641	18.49621	-124.343	-37.2909
4	-1.71442	-0.24671	0.56222	-39.7175	19.26043	8.8618
5	-0.02546	-0.3302	0.8818	-12.6488	-25.3045	-6.87866
6	-1.62165	-0.24768	0.49728	-2.78623	-9.27392	4.61753
7	-1.71442	-0.24671	0.56222	-18.6183	20.03905	8.8618
8	0.0324	-0.25452	1.32271	-4.81676	6.49707	4.26638
9	-0.13459	-0.10218	0.56832	-28.5114	-7.77774	-8.35957
10	-1.62165	-0.35413	0.56319	13.74812	7.96272	41.96104
Nilai terendah	-1.71442	-0.35413	0.35641	-39.7175	-124.343	-37.2909
Nilai tertinggi	0.0324	-0.10218	1.32271	18.49621	20.03905	41.96104

3.2. Rancangan Sistem

Penelitian akan membuat sebuah sistem *animal tracking* berbasis *Iot* yang mampu melacak keberadaan dan keadaan hewan ternak serta membangun suatu *geo-fencing* pada area peternakan. Untuk mewujudkannya diperlukan adanya sebuah perangkat yang berperan sebagai *master(gateway)* dan beberapa perangkat yang berperan sebagai *client(node)*. Dalam penelitian ini, jumlah *client* yang digunakan adalah 2 buah. Antara *master* dan *client* dihubungkan dengan jaringan *LoRa* sehingga bisa berkomunikasi tanpa jaringan seluler atau *WiFi*. Untuk komunikasi antara *master* dan *firebase*, digunakan jaringan *WiFi*. Kemudian data tersebut bisa diakses melalui aplikasi *Kodular* dalam bentuk marker pada map beserta nilai *latitude* dan *longitude* dari hewan ternak tersebut. Gambar 12 menunjukkan alur data dari sistem *animal tracking* dan Gambar 13 memperlihatkan blok diagram *animal tracking* secara keseluruhan.



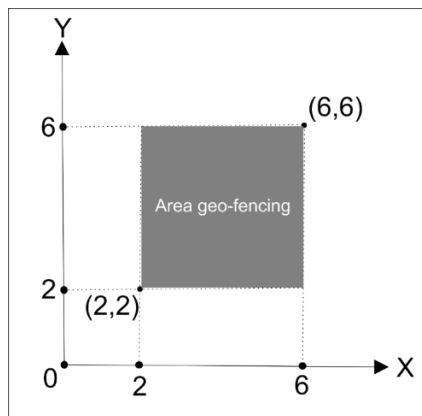
Gambar 12. Alur data sistem animal tracking berbasis Iot



Gambar 13. Blok diagram sistem animal tracking berbasis Iot

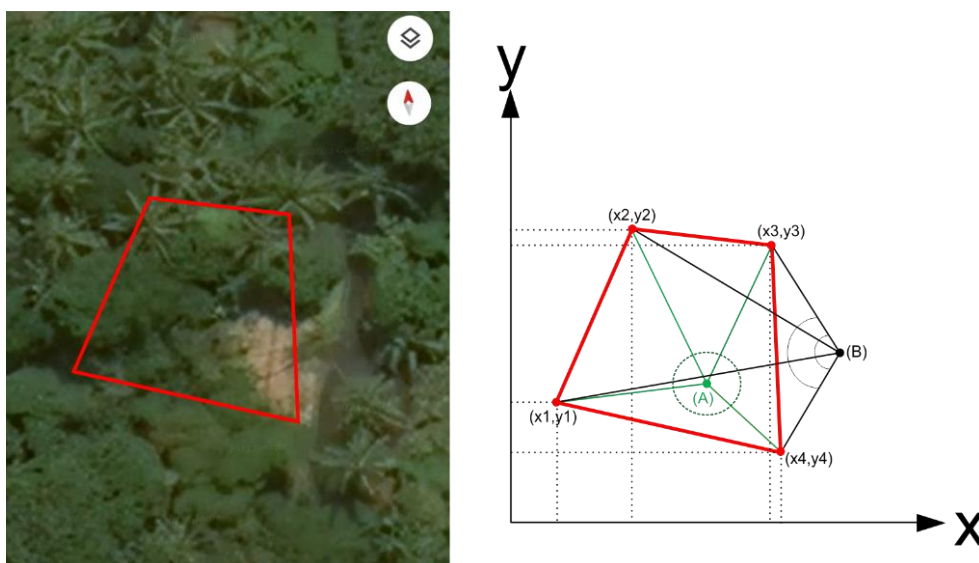
3.2.1. Rancangan *Geo-fence*

Geo-fencing bisa dibangun dengan memanfaatkan sistem *GPS*. Sistem *GPS* memetakan bumi dalam 2 sumbu yaitu *latitude* dan *longitude*. *Latitude* dan *longitude* ini sama halnya dengan sumbu x dan y pada matematika. Oleh karena itu, untuk membuat sebuah bangun datar berbentuk persegi bisa dilakukan cukup dengan menandai 2 titik saja. Suatu objek akan dianggap keluar dari area yang ditentukan apabila nilai x atau y -nya melebihi batas atas atau kurang dari batas bawah yang telah ditentukan. Untuk lebih jelasnya bisa dilihat pada Gambar 14 Konsep membangun *geo-fencing*.



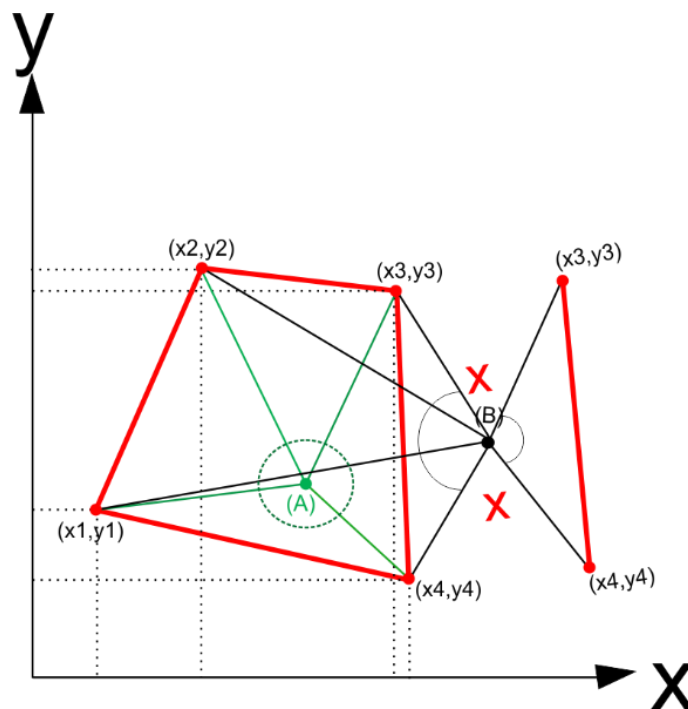
Gambar 14. Konsep membangun *Geo-fence*

Penelitian ini menggunakan area pengambilan data yang sama dengan keadaan di lapangan. Sehingga bentuk area *Geo-fence* juga tidak sesuai dengan sumbu *latitude* ataupun *longitude*. Gambar 15 menunjukkan bentuk lahan peternakan melalui app google maps dan area peternakan pada sumbu x dan y .



Gambar 15. Bentuk lahan peternakan

Area peternakan yang digunakan berbentuk segi empat sembarang. Namun bagaimanapun bentuk ini merupakan suatu bangun datar dan setiap bangun datar (kecuali segitiga) memiliki sudut total sebesar 360° . Apabila kita membuat satu titik baik itu diluar ataupun didalam suatu bangun datar (kecuali bangun datar tidak bersudut) kemudian titik itu kita hubungkan ke tiap-tiap sudut bangun datar maka akan didapat beberapa segitiga. Pada Gambar 15 terdapat titik A (didalam) dan titik B (diluar), kedua titik ini sudah terhubung dengan sudut-sudut bangun datar dan membentuk masing-masing 4 buah segitiga. Apabila sudut-sudut segitiga yang berada di titik A dijumlahkan maka akan didapat nilai sudut sebesar 360° (lingkaran hijau disekitar titik A). Sedangkan pada titik B, apabila sudut-sudut segitiga yang berada di titik B dijumlahkan, hasil yang didapat tidak akan pas 360° . Apabila segitiga (x_3, y_3) , (B), (x_4, y_4) kita balik secara vertikal maka akan terlihat sudut total yang terbentuk tidak sama dengan 360° . Gambar 16 segitiga (x_3, y_3) , (B), (x_4, y_4) dibalik secara vertikal.

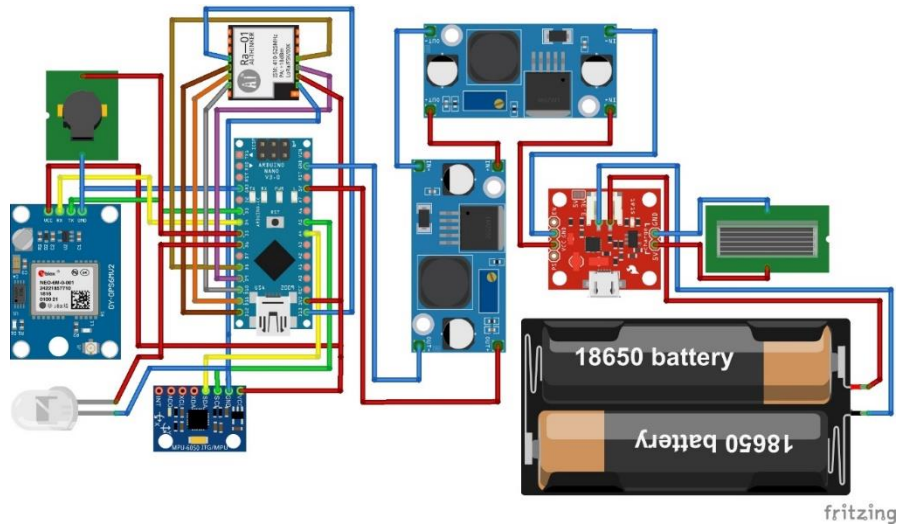


Gambar 16. Bentuk lahan saat segitiga (x_3, y_3) , (B), (x_4, y_4) diputar pada sumbu vertikal

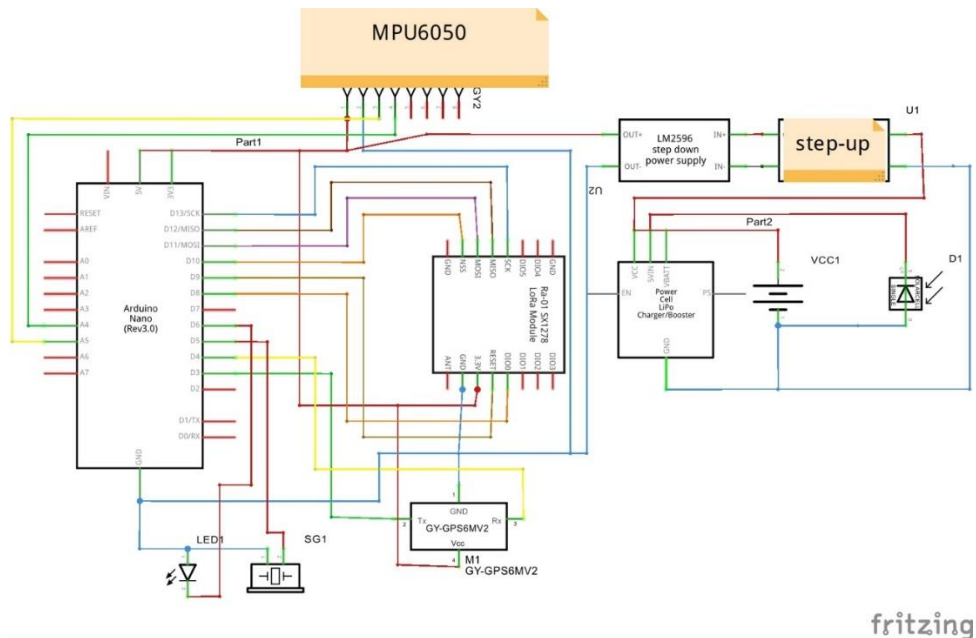
Berdasarkan Gambar 9, terlihat ada area yang berisi x merah. Area tersebut adalah area yang hilang dari perhitungan yang menyebabkan total sudut pada titik B tidak akan sama dengan 360° . Untuk mengetahui suatu sudut segitiga dapat menggunakan persamaan trigonometri dengan catatan mengetahui panjang dari sisi-sisi segitiga. Panjang anatar titik bisa didapatkan secara otomatis oleh sistem *GPS* hal ini juga yang dimanfaatkan Google Maps untuk mengetahui jarak antara pengguna dengan titik lokasi.

3.2.2. Rancangan Perangkat *Client*

Perangkat *Client* berfungsi sebagai pemindai lokasi dan keadaan dari hewan ternak. Data lokasi dan keadaan yang didapat kemudian diteruskan ke *master* melalui jaringan *LoRa*. Untuk dapat mewujudkannya, diperlukan sebuah mikrokontroler, sebuah modul *GPS*, sebuah modul sensor MPU6050 dan sebuah *transmitter LoRa*. Untuk lebih jelasnya bisa dilihat pada gambar 17 *Wiring diagram* perangkat *client* dan gambar 18 *Skematik* perangkat *client*.



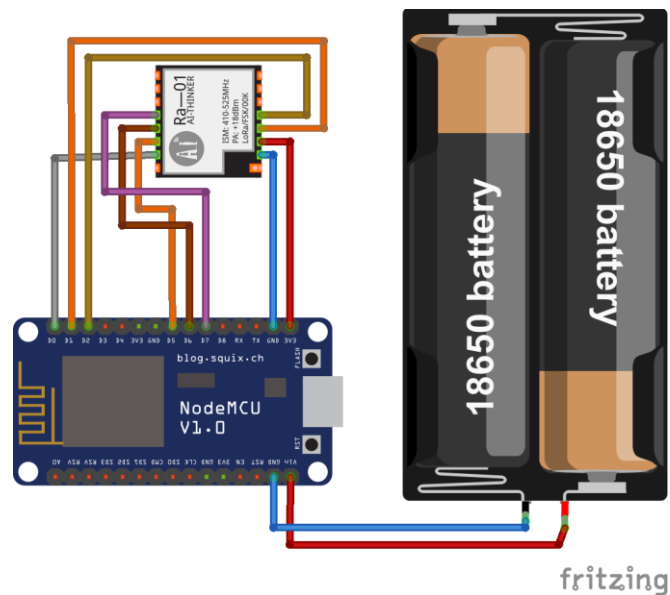
Gambar 17. *Wiring diagram* perangkat *client*



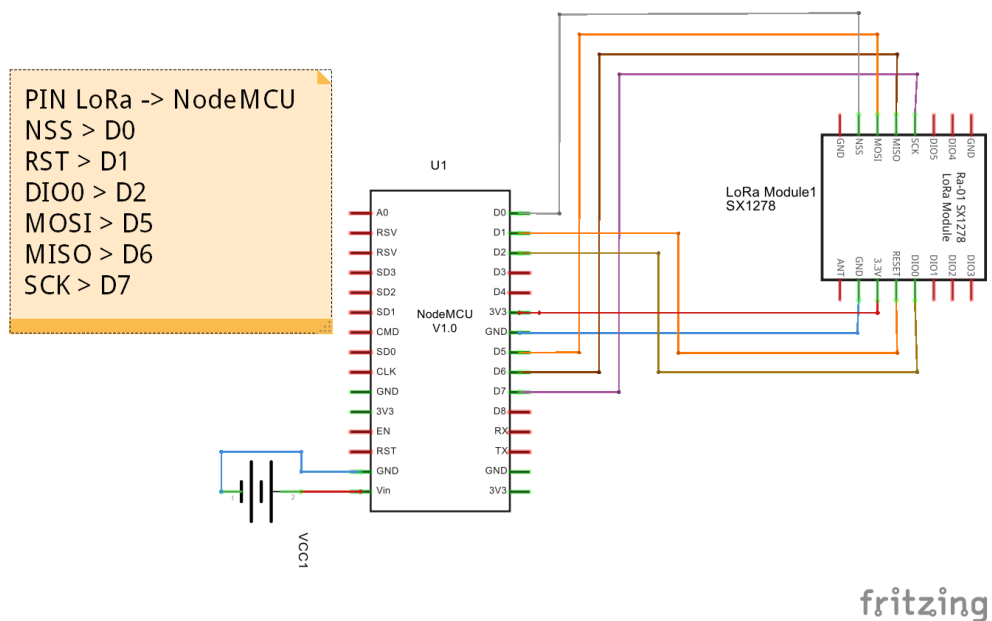
Gambar 18. *Skematik* perangkat *client*

3.2.3. Rancangan Perangkat *Master*

Perangkat *master* berfungsi sebagai *Gateway* dari sistem *animal tracking* ke firebase melalui jaringan *WiFi*. Untuk dapat menerima data dari *client* dan mengirimkannya ke firebase, diperlukan sebuah *receiver LoRa* untuk berkomunikasi dengan *client* dan sebuah mikrokontroler yang mampu terhubung dengan *WiFi* agar dapat berkomunikasi dengan firebase. Untuk lebih jelasnya bisa dilihat pada gambar 19 *Wiring diagram* perangkat *master* dan gambar 20 *Skematik* perangkat *master*.



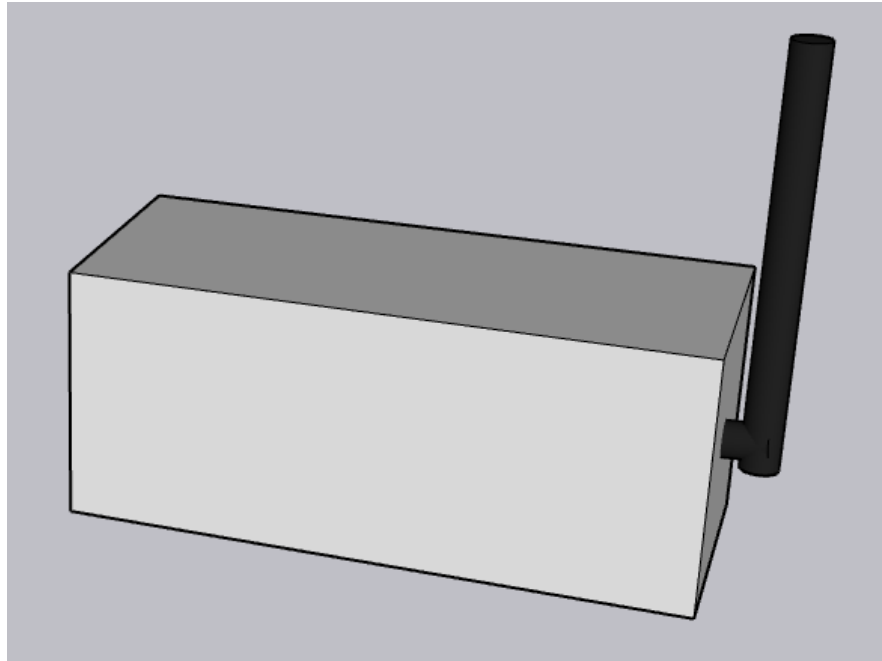
Gambar 19. *Wiring diagram* perangkat *master*



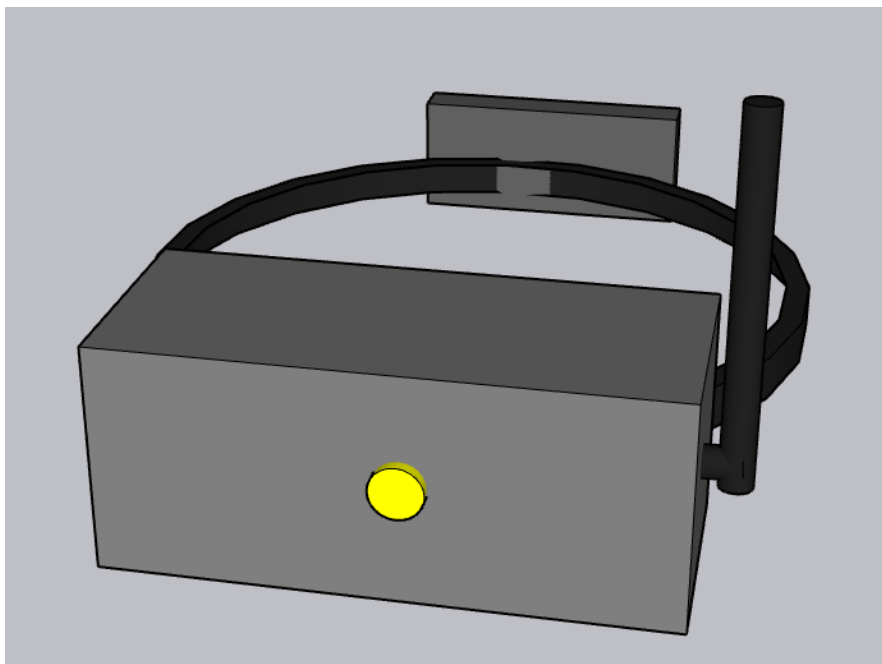
Gambar 20. *Skematik* perangkat *master*

3.2.4. Rancangan Casing Komponen

Untuk melindungi komponen dari kemungkinan kerusakan fisik dari benturan ataupun zat-zat berbahaya, diperlukan sebuah casing yang mampu memberi perlindungan bagi komponen-komponen tersebut. Gambar 21 menunjukkan desain casing cari perangkat master, sedangkan Gambar 22 menunjukkan desain casing perangkat client.



Gambar 21. Desain perangkat master



Gambar 22. Desain perangkat client

3.3. Pembuatan Alat/Implementasi Sistem

3.3.1. Tahap Pembuatan *Hardware*

Pembuatan *hardware* diawali dengan pengumpulan alat. Setelah semua bahan terkumpul, dilanjutkan dengan tahap pengujian kerja alat. Pengujian ini dilakukan untuk memastikan bahwa alat yang sudah didapat bisa berfungsi dengan baik. Apabila semua alat sudah dites dan berfungsi dengan baik, dilanjutkan dengan tahap perakitan. Perakitan dilakukan sesuai dengan rancangan yang disebutkan pada poin 3.1. rancangan sistem diatas. Alat dan bahan yang digunakan antara lain:

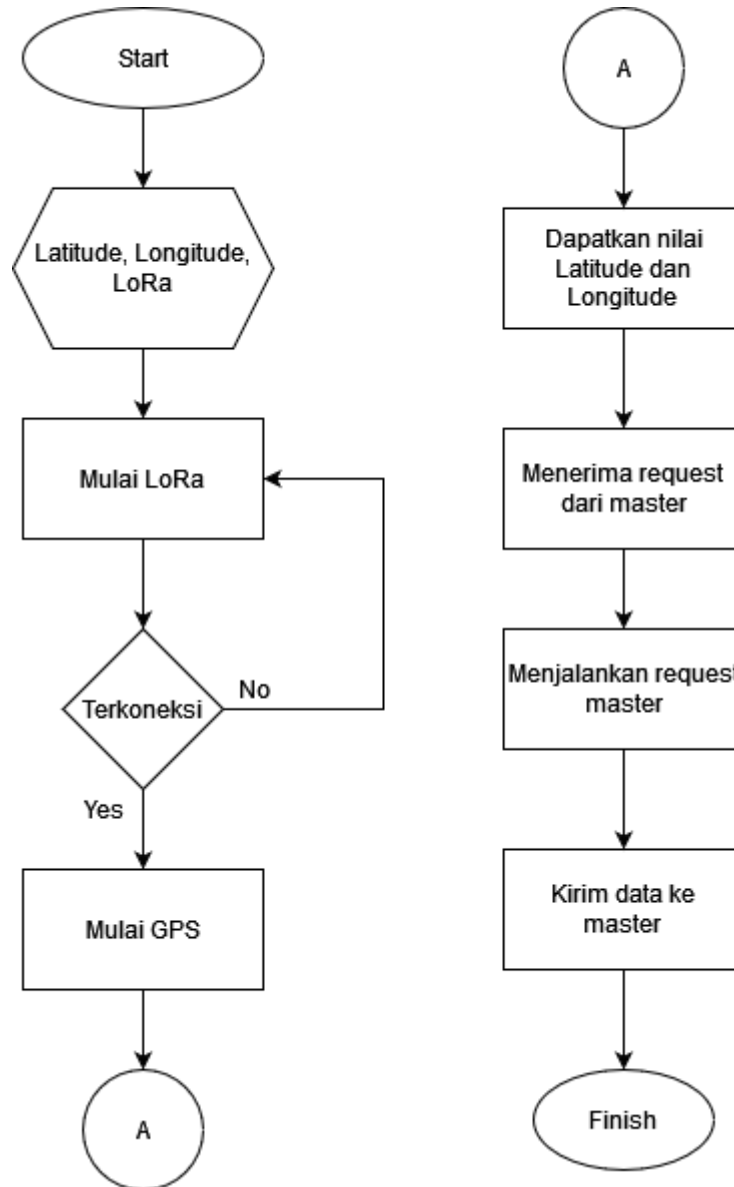
- a. NodeMCU ESP8266 sebanyak 1 buah
- b. Arduino Nano sebanyak 2 buah
- c. Modul *transceiver LoRa* RA-02 SX1276 sebanyak 3 buah
- d. Modul *GPS* Neo-6M sebanyak 2 buah
- e. Modul sensor MPU6050 sebanyak 2 buah
- f. Modul *battery charger* sebanyak 2 buah
- g. Modul DC-DC *step up* sebanyak 2 buah
- h. Modul DC-DC *step down* sebanyak 2 buah
- i. *Buzzer* sebanyak 2 buah
- j. *LED* sebanyak 2 buah
- k. Panel surya mini 5V sebanyak 2 buah
- l. Baterai 18650 sebanyak 6 buah
- m. 18650 *battery holder 2 slot* sebanyak 3 buah
- n. Kabel penghubung secukupnya

3.3.2. Tahap Pembuatan *Software*

Perangkat *Client*, perangkat *master* dan aplikasi memiliki *software* yang berbeda. Meskipun *software*-nya berbeda, ketiganya harus mampu saling berkomunikasi agar seluruh sistem dapat bekerja dengan baik. Perangkat *client* dan *master* akan diprogram menggunakan aplikasi Arduino IDE, untuk *database* menggunakan Firebase *database*,

sedangkan untuk aplikasi akan dibuat melalui *website* creator.Kodular. Berikut adalah *flowchart* dari perangkat *client*, perangkat *master* serta aplikasi.

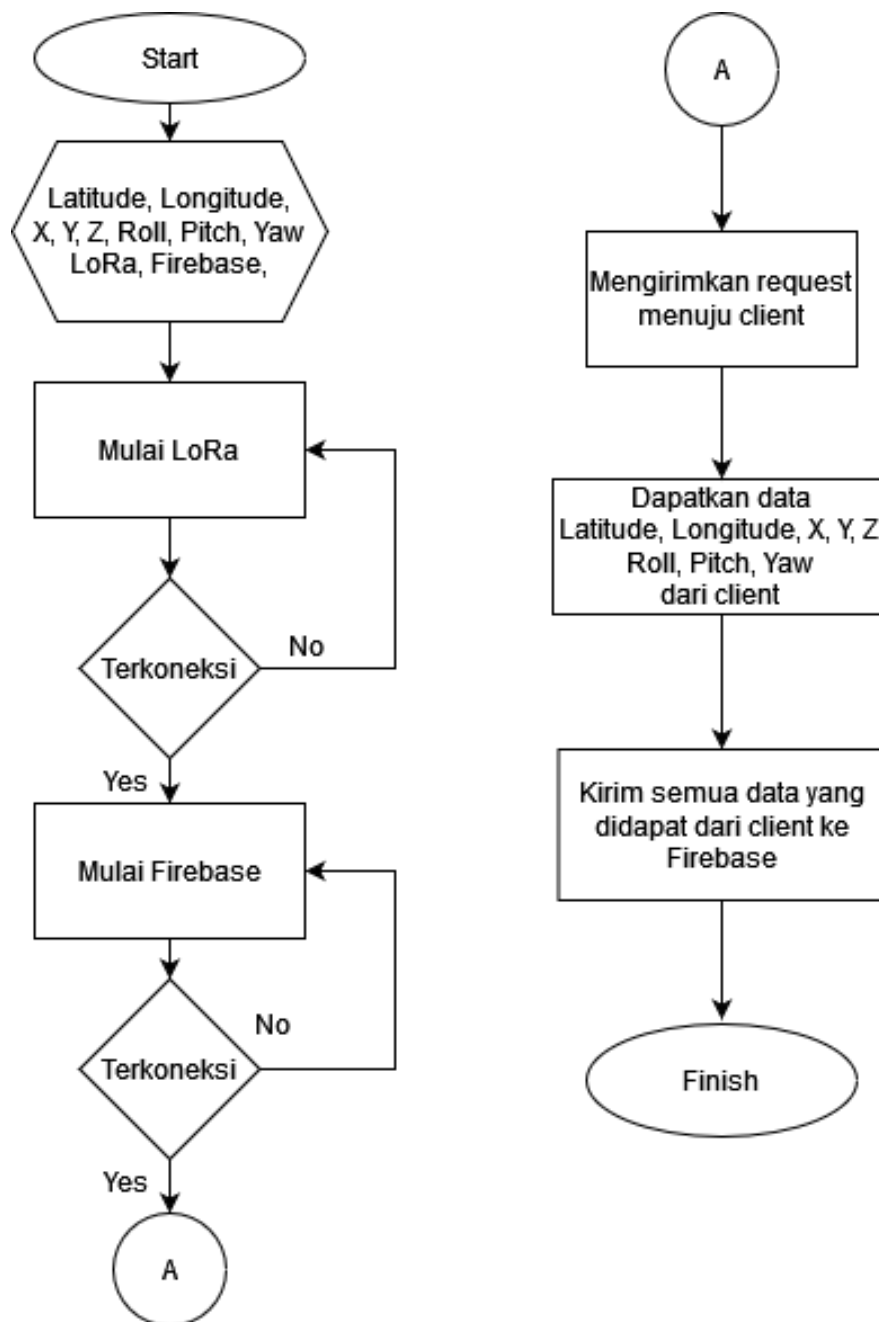
a. *Flowchart* perangkat *client*



Gambar 23. *Flowchart* perangkat *client*

Saat perangkat *client* dimulai, diawali dengan inisialisasi semua variabel yang diperlukan. Kemudian dilanjutkan dengan memulai *LoRa*, apabila gagal proses ini akan diulangi. Setelah itu dilanjutkan dengan memulai *GPS* untuk mendapatkan nilai *latitude* serta *longitude*, kemudian memulai MPU6050 untuk mendapatkan nilai *x*, *y*, *z*, *roll*, *pitch* dan *yaw*. Kemudian dilanjutkan dengan merespon *request* yang diberikan oleh perangkat *master*.

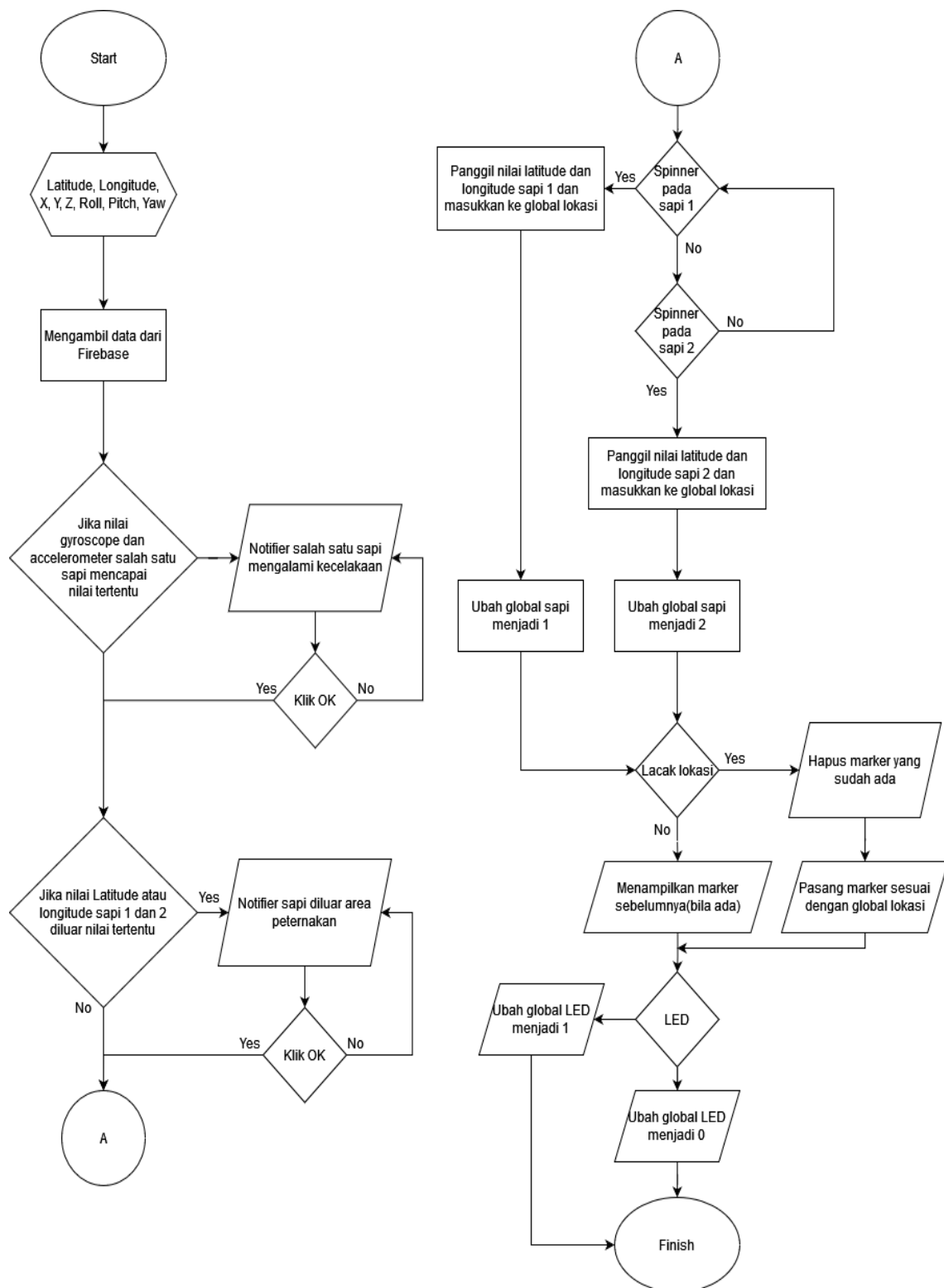
b. *Flowchart perangkat master*



Gambar 24. *Flowchart perangkat master*

Saat perangkat *master* dimulai, diawali dengan inisialisasi semua variabel yang diperlukan. Kemudian dilanjutkan dengan memulai *LoRa* dan *Firebase*, apabila gagal kedua tahap tersebut akan diulangi. Setelah itu dilanjutkan dengan mengirim *request* kepada perangkat *client* dan menerima data pembacaan sensor dari perangkat *client*. Kemudian semua data yang sudah didapat dikirimkan ke *firebase*.

c. Flowchart aplikasi



Gambar 25. Flowchart aplikasi

Setelah aplikasi dimulai, dilakukan inisialisasi semua variabel kemudian dilanjutkan dengan mengambil data dari firebase. Apabila nilai gyroscope dan accelerometer salah satu client mencapai batas tertentu maka akan memunculkan notifikasi ternak mengalami

kecelakaan, apabila tidak maka dilanjutkan menuju proses selanjutnya. Dalam notifikasi ada tombol OK untuk melanjutkan ke proses berikutnya. Selanjutnya apabila nilai *latitude* dan *longitude* tidak sesuai dengan nilai-nilai tertentu maka akan muncul notifikasi ternak keluar area peternakan, apabila tidak maka dilanjutkan menuju proses selanjutnya. Dalam notifikasi ini juga ada tombol OK untuk melanjutkan ke proses berikutnya. Apabila spinner pada sapi 1 maka *latitude* dan *longitude* sapi satu dimasukkan ke *global* lokasi kemudian mengubah *global* sapi menjadi 1, apabila tidak maka dilanjutkan pada proses berikutnya. Apabila spinner pada sapi 2 maka *latitude* dan *longitude* sapi dua dimasukkan ke *global* lokasi kemudian mengubah *global* sapi menjadi 2, apabila tidak maka proses kembali pada pemilihan spinner. Apabila tombol lacak lokasi ditekan maka akan menghapus marker sebelumnya dan memasang marker baru sesuai dengan *global* lokasi, apabila tidak maka akan menampilkan marker yang sebelumnya sudah ada. Apabila tombol *LED* ditekan maka akan mengubah *global LED* menjadi 1, apabila tidak maka *global LED* tetap 0.

3.4. Pengujian

3.4.2. Pengujian Kemampuan Melacak Lokasi

Pengujian ini dilakukan untuk mengetahui seberapa akurat kemampuan lacak lokasi dari sistem *animal tracking*. Pengujian dilakukan dengan menempatkan perangkat *client* bersebelahan dengan *smartphone* yang sudah terinstal aplikasi Google Maps.

Data yang diambil adalah nilai *latitude* dan *longitude* dari sistem *animal tracking* sebagai sampel dan aplikasi Google Maps sebagai acuan. Selanjutnya nilai sampel dan acuan akan dibandingkan untuk mendapat presentase selisih dari keduanya. Persamaan yang digunakan untuk menghitung presentase selisih adalah sebagai berikut.

$$Latitude = \frac{|(Nilai latitude acuan - sampel)|}{|Nilai latitude acuan|} \times 100\% \dots \dots \dots (1)$$

$$Longitude = \frac{|(Nilai longitude acuan - sampel)|}{|Nilai longitude acuan|} \times 100\% \dots \dots \dots (2)$$

Setelah didapat presentase selisih dari *latitude* dan *longitude*, dilanjutkan dengan meratakannya untuk mendapatkan nilai akurasi secara keseluruhan. Persamaan yang digunakan adalah sebagai berikut.

$$Rata-rata = \frac{\sum x}{n} \dots \dots \dots (3)$$

Dimana:

- Σx = jumlah nilai semua data
- n = banyaknya data

3.4.3. Pengujian Kemampuan Memindai Gerakan

Pengujian ini dilakukan setelah mendapatkan semua dataset gerakan sapi. Pengujian ini bertujuan untuk menguji keandalan sistem pelacak gerakan hewan. Pengujian ini dilakukan dengan melakukan simulasi gerakan hewan kemudian melihat ketepatan respon dari sistem serta waktu *delay* antara kejadian dengan notifikasi pada aplikasi. Data waktu *delay* akan dirata-ratakan menggunakan persamaan (3). Sedangkan data ketepatan respon akan diolah guna mendapatkan presentase keberhasilan. Untuk bisa mendapatkan presentase keberhasilan digunakan persamaan sebagai berikut.

$$\text{Presentase keberhasilan} = \frac{\text{Jumlah respon tepat}}{\text{Jumlah percobaan}} \times 100\% \dots \dots \dots (4)$$

3.4.4. Pengujian Keandalan *Geo-fencing*

Pengujian ini menguji keandalan *geo-fencing* dalam mengamankan hewan ternak. Serta ketepatan sistem dalam memindai pergerakan temporal hewan ternak. Pengujian keamanan dilakukan dengan memindahkan perangkat *client* hingga keluar dari area *geo-fencing*. Jarak dari perangkat *client* keluar dari area *geo-fencing* hingga muncul notifikasi pada aplikasi Kodular akan diambil sebagai sampel data eror. Pengujian ini akan dilakukan beberapa kali. Semua data yang didapat akan dirata-ratakan menggunakan persamaan (3) sehingga didapat nilai akurasi dari sistem *geo-fencing*.

Pengujian ketepatan memindai gerakan temporal dilakukan dengan memasukkan perangkat *client* ke dalam area-area yang sudah dibuat, antara lain area istirahat dan area merumput. Data yang diambil adalah keberhasilan memindai sehingga digunakan persamaan (4) untuk menghitung presentase keberhasilan.

3.4.5. Pengujian Pengaruh Jarak Terhadap Kekuatan Sinyal *LoRa*

Pengujian akan menguji pengaruh jarak terhadap kekuatan sinyal *LoRa*. Parameter utama dalam pengujian ini adalah nilai *RSSI* dan jarak. Pengujian ini dilakukan dengan memisahkan perangkat *master* dan *client* pada jarak tertentu kemudian mengamati perubahan pada nilai *RSSI*. Penelitian ini akan dilakukan beberapa kali. Data *RSSI* dan jarak akan dibandingkan dalam bentuk grafik sehingga didapat tren yang terjadi

BAB IV HASIL DAN PEMBAHASAN

4.1. Hasil

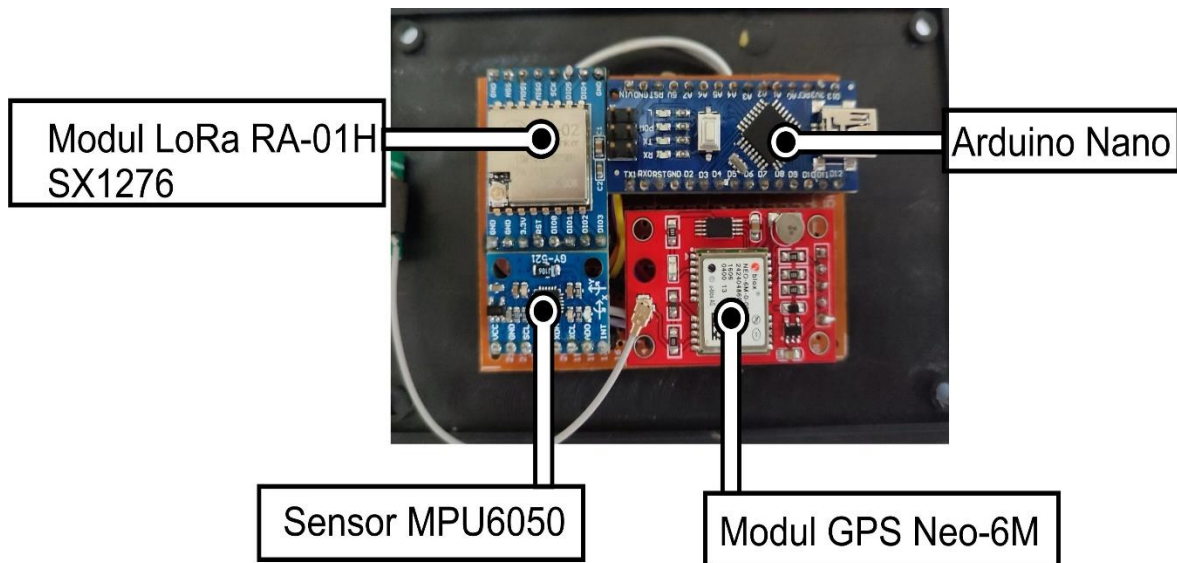
Penelitian ini telah berhasil membuat sistem *animal tracking* berbasis *IOT*. Adapun hasil yang didapat adalah sebagai berikut.

4.1.1. Alat *Animal Tracking* Berbasis *IOT*

Penelitian ini membuat tiga buah alat. Dua alat merupakan perangkat *client* yang dan satu lagi adalah perangkat *master*.

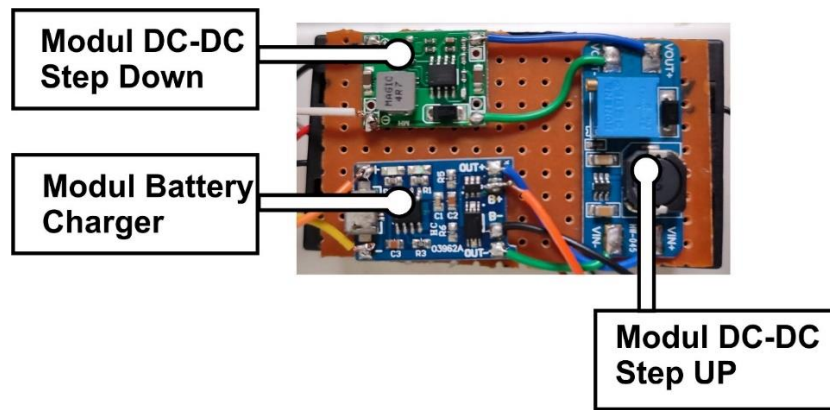
4.1.1.1. *Hardware*

Perangkat *client* terbagi menjadi dua komponen utama yaitu bagian kontrol dan bagian *battery charging*. Bagian kontrol terdiri dari Arduino nano, modul *LoRa*, Sensor MPU dan Modul *GPS*. Gambar 26 menunjukkan komponen dari bagian kontrol, sedangkan Gambar 27 menunjukkan komponen dari bagian *battery charging* perangkat *client*.



Gambar 26. Komponen bagian kontrol perangkat client

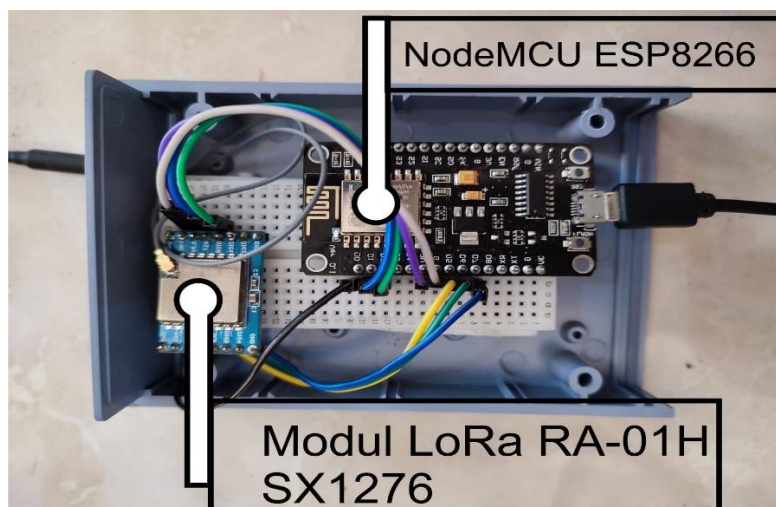
Bagian kontrol disusun sedemikian rupa hingga didapat bentuk paling minimalis. Sehingga didapat dimensi total dari bagian ini adalah panjang 7cm, lebar 5cm dan tinggi 2,5cm dengan bobot total sebesar 50gr.



Gambar 27. *Komponen bagian battery charging perangkat client*

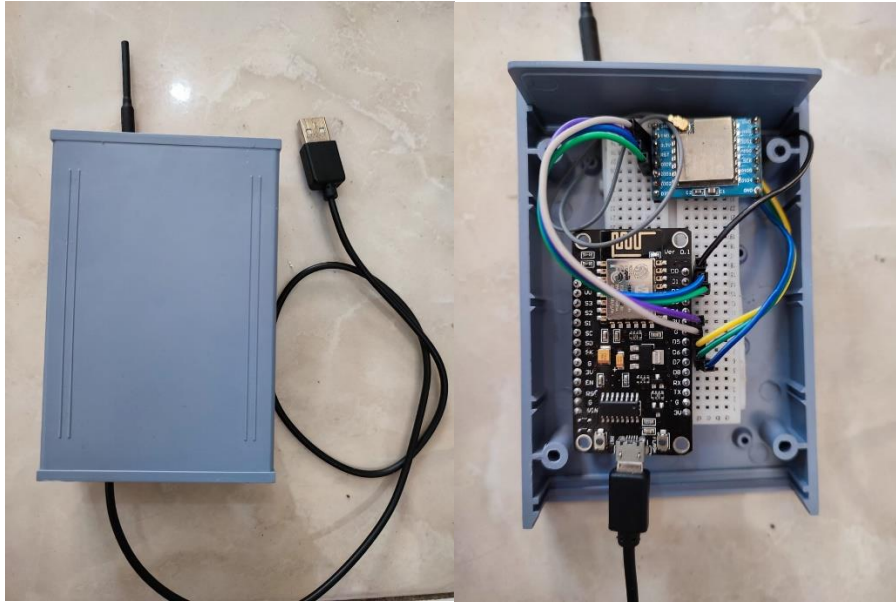
Untuk bagian *battery charging* terdiri dari modul *battery charger*, modul DC-DC *step up* dan modul DC-DC *step down*. Komponen-komponen tersebut disusun seperti pada Gambar 27. Bagian ini memiliki dimensi panjang 6,5cm, lebar 4cm dan tinggi 1cm dengan bobot total 25gr.

Perangkat *master* terdiri dari dua komponen utama yaitu nodeMCU ESP8266 sebagai mikrokontroler Modul *LoRa* sebagai media transfer data. Gambar 28 menunjukkan komponen-komponen perangkat *master*.



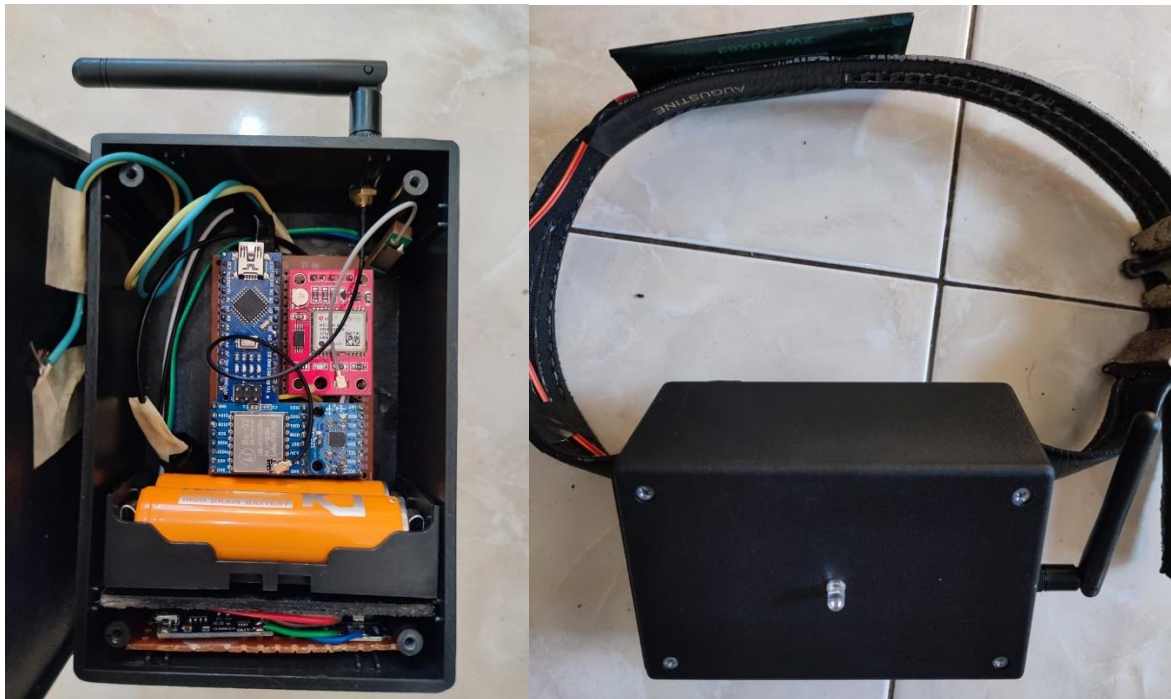
Gambar 28. *Komponen-komponen perangkat master*

Perangkat *master* memiliki dimensi panjang 11cm, lebar 8cm dan tinggi 5cm dengan bobot total 150gr. Gambar 29 menunjukkan perangkat *master animal tracking* berbasis *IOT*.



Gambar 29. Perangkat master animal tracking berbasis IOT

Perangkat *client* berdimensi panjang 13cm, lebar 8cm, tinggi 5cm dan tali pengikat sepanjang 90cm dengan bobot total 250gr. Gambar 30 menunjukkan perangkat *client* animal tracking berbasis IOT



Gambar 30. Perangkat client animal tracking berbasis IOT

4.1.1.2. Software

Agar alat *animal tracking* berbasis *IOT* ini dapat bekerja sesuai dengan rancangan yang sudah dibuat, diperlukan *software* yang sesuai untuk menjalankan semua fitur yang sudah direncanakan.

Untuk menjalankan *LoRa*, *WiFi* dan *firebase* pada perangkat *master*, digunakan library *SPI.h* dan *LoRa* sebagai media interfacing *LoRa* dengan mikrokontroler. Selain itu, adapun beberapa library *SoftwareSerial.h* dan *ESP8266WiFi* untuk menghubungkan mikrokontroler dengan *WiFi*. *FirebaseArduino.h* digunakan untuk menghubungkan mikrokontroler dengan *firebase database*. Pada perangkat *client* juga menggunakan *SPI.h* dan *LoRa.h*. selain itu pada perangkat *client* juga menggunakan library *TinyGPS++.h* untuk menjalankan modul *GPS*. *Wire.h* dan *MPU6050_light.h* untuk interfacing *MPU6050* ke mikrokontroler melalui pin *I2C*. Gambar 31 menunjukkan include library perangkat *master* dan include library perangkat *client*.

```
1 #include <SPI.h>
2 #include <LoRa.h>
3 #include <SoftwareSerial.h>
4 #include <ESP8266WiFi.h>
5 #include <FirebaseArduino.h>

1 #include <SPI.h>
2 #include <LoRa.h>
3 #include <TinyGPS++.h>
4 #include <SoftwareSerial.h>
5 #include <Wire.h>
6 #include <MPU6050_light.h>
```

Gambar 31. Include library perangkat *master* dan *client*

Setelah mendefinisikan library yang digunakan, dilanjutkan dengan memulai *setup* dari library yang sudah didefinisikan. Pada perangkat *master* diawali dengan memulai serial pada baud 115200. Kemudian dilanjutkan dengan memulai *WiFi*, proses ini akan diulangi hingga dapat sukses terhubung ke *WiFi*. Setelah terhubung dengan *WiFi*, dilanjutkan dengan memulai *firebase* untuk menghubungkan mikrokontroler ke *firebase database*. Saat *firebase* dan mikrokontroler sudah terhubung, dilanjutkan dengan memulai *LoRa*.

Perangkat *client* juga menggunakan *LoRa*, namun tidak menggunakan *WiFi* dan *firebase*. Pada perangkat *client* diawali dengan memulai serial, kemudian memulai *wire.h* untuk menghubungkan *MPU6050* melalui pin *I2C*. Setelah itu dilanjutkan dengan memulai *LoRa*. Kemudian setelah *MPU6050* terhubung melalui pin *I2C*, dilanjutkan dengan memulai kerja *MPU6050*. Gambar 32 menunjukkan *void setup* perangkat *master* dan *void setup* perangkat *client*.

```

void setup() {
  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
    Serial.println("");
  }
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Serial.println("LoRa Master Node");
  LoRa.setPins(nss, rst, dio0);
  if (!LoRa.begin(915E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  Serial.print("LoRa Online");
}

void loop() {
  Serial.println("");
  Serial.println("LoRa Node1");
  LoRa.setPins(nss, rst, dio0);
  if (!LoRa.begin(915E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  Serial.println("LoRa Online");
  ss.begin(GPSBaud);
  byte status = mpu.begin();
  Serial.print(F("MPU6050 status: "));
  Serial.println(status);
  while(status!=0) {
    Serial.println(F("Calculating offsets, do not move MPU6050"));
    delay(1000);
    mpu.calcOffsets(true,true);
    Serial.println("Done!\n");
  }
}

```

Gambar 32. Void setup perangkat master dan client

Pengiriman data melalui *LoRa* diatur sepenuhnya oleh perangkat *master*. Perangkat *master* akan mengatur lalu lintas data dari dua *client* dengan mengatur *timing* pengiriman. Perangkat *client* hanya akan mengirimkan data melalui *LoRa* apabila diminta oleh perangkat *master*. Perangkat *master* akan mengirimkan *request* secara bergantian dan *client* akan membalas secara bergantian pula. Dengan mengatur *timing request* pengiriman maka data yang masuk juga akan teratur. Gambar 33 menunjukkan salah satu *coding timing* pengiriman.

```

if ((unsigned long)(currentsecs - previoussecs) >= interval) {
  Secs = Secs + 1;
  //Serial.println(Secs);
  if ( Secs >= 4 )
  {
    Secs = 0;
  }

  if ( (Secs >= 0) && (Secs < 1) && (lamp1 == 0))
  {
    String message = "10";

```

Gambar 33. Coding timing pengiriman

Apabila perangkat *master* sudah menerima data dari salah satu *client*, data tersebut akan dikirimkan ke firebase *database* melalui jaringan *WiFi*. Gambar 34 adalah *coding* pengiriman data ke firebase *database*.


```

Firebase.setFloat("latitudel", latitudel);
Firebase.setFloat("longitudel", longitudel);
Firebase.setFloat("accelerationl", accelerationl);
Firebase.setFloat("gyroscopel", gyroscopel);
Firebase.setInt("fencel", GFencel);
delay(200);

```

Gambar 34. Coding pengiriman data ke firebase

Setiap kali perangkat *client* menerima *request* dari perangkat *master*, perangkat *client* akan mengirimkan data melalui *LoRa*. Selain mengirimkan data, perangkat *client* juga akan melakukan tugas lain sesuai dengan isi dari pesan *request*. Gambar 35 merupakan *coding* salah satu eksekusi dari *request* yang dikirimkan perangkat *master*.

```

if (Val == 10)
{
  Mymessage = Mymessage + latitudel+" "+longitudel+" "+ACCEL1+" "+Gyrol+" "+targetStatus;
  sendMessage(Mymessage, MasterNode, Nodel);
  delay(100);
  Mymessage = "";
  digitalWrite(Lamp, LOW);
}

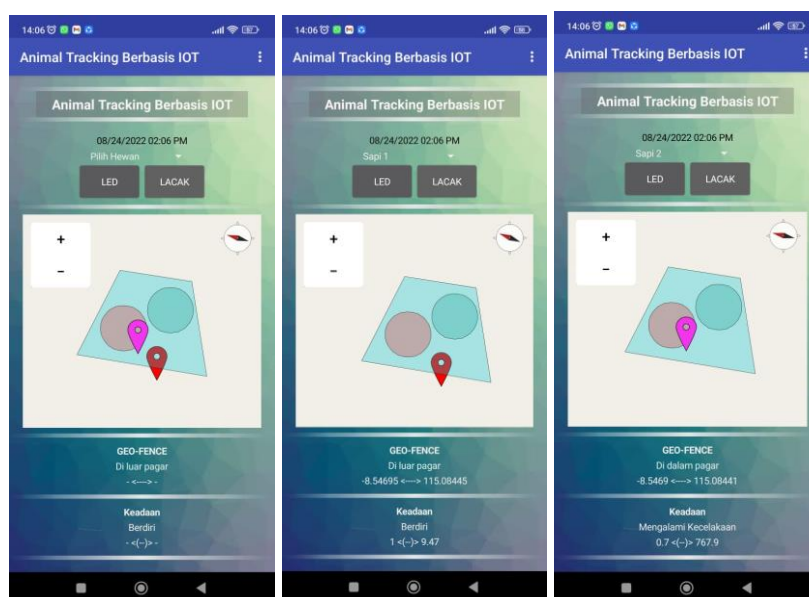
```

Gambar 35. Coding eksekusi request

4.1.2. Aplikasi *Animal Tracking* Berbasis *IOT*

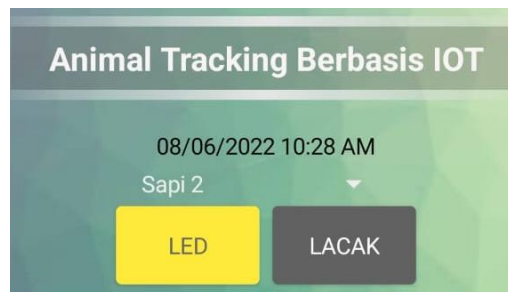
Penelitian ini juga membuat sebuah aplikasi berbasis android yang berfungsi sebagai media monitoring lokasi dan keadaan hewan ternak, serta sebagai remote kontrol nyala *LED* pada alat *animal tracking*.

4.1.2.1. Aplikasi

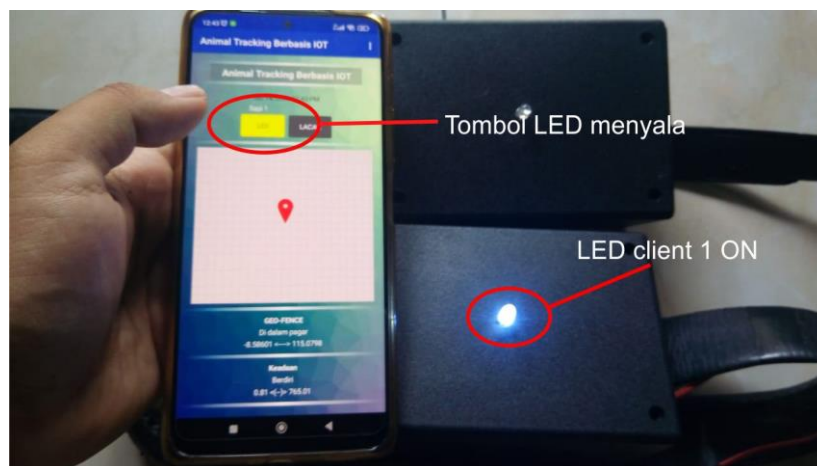


Gambar 36. Aplikasi *animal tracking* berbasis *Iot*

Dalam aplikasi berisi spinner untuk memilih sapi mana yang mau dilacak lokasi ataupun keadaannya. Dalam spinner terdapat dua pilihan yaitu sapi 1 dan sapi 2. Aplikasi ini juga berisi dua buah tombol yaitu *LED* dan *LACAK*. Tombol *LACAK* berfungsi untuk melacak keberadaan dan keadaan sapi yang dipilih pada spinner. Tombol *LED* berfungsi untuk mengatur nyala *LED* pada alat *animal tracking*. Apabila tombol *LED* belum ditekan maka akan berwarna abu-abu dan *LED* pada alat dalam kondisi OFF. Apabila ditekan sekali tombol *LED* akan berwarna kuning dan *LED* pada alat menyala. Apabila ditekan untuk kedua kali maka *LED* pada alat akan mati dan tombol kembali berwarna abu-abu. Gambar 37 menunjukkan tombol *LED* saat ditekan sekali sedangkan Gambar 38 menunjukkan reaksi dari perangkat *client* saat tombol *LED* ditekan sekali.

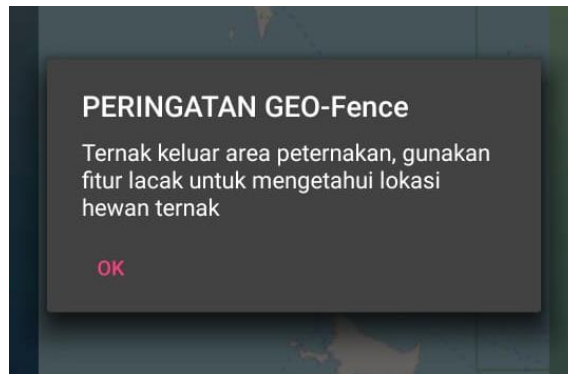


Gambar 37. Tampilan saat tombol *LED* ditekan sekali

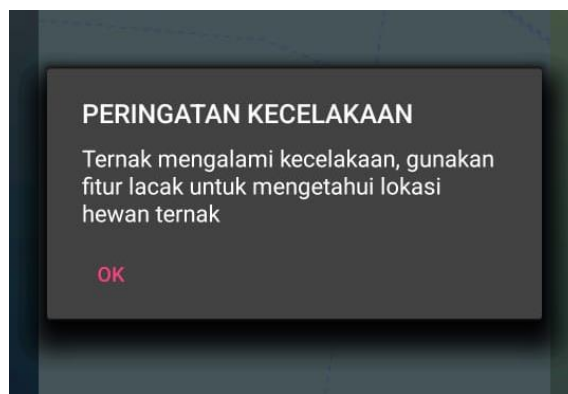


Gambar 38. Reaksi dari perangkat *client* saat tombol *LED* ditekan sekali

Apabila hewan ternak mengalami kecelakaan ataupun meninggalkan area peternakan, aplikasi akan menampilkan pemberitahuan yang langsung pop up di layar. Gambar 39 menunjukkan pemberitahuan ternak keluar area peternakan dan Gambar 40 menunjukkan pemberitahuan ternak mengalami kecelakaan.



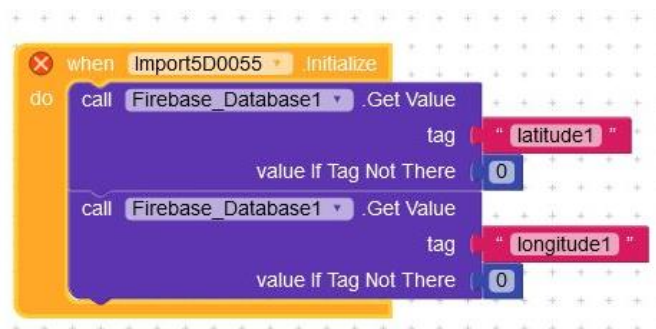
Gambar 39. Pemberitahuan ternak keluar area peternakan



Gambar 40. Pemberitahuan ternak mengalami kecelakaan

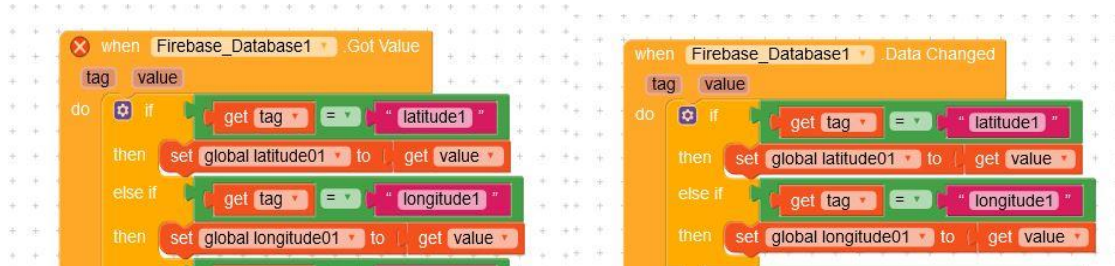
4.1.2.2. Coding

Agar bisa mengambil data dari firebase serta mengolahnya hingga bisa ditampilkan pada layar aplikasi, perlu dibuat *coding* untuk merealisasikannya. Pada website Koular disediakan media koding berupa *block* program. Untuk memanggil firebase digunakan perintah call firebase *database* yang dipasang pada frame screen initialize. Dengan demikian pada saat screen baru dimulai aplikasi akan langsung memanggil firebase. Gambar 41 adalah frame untuk memanggil firebase saat layar dimulai.



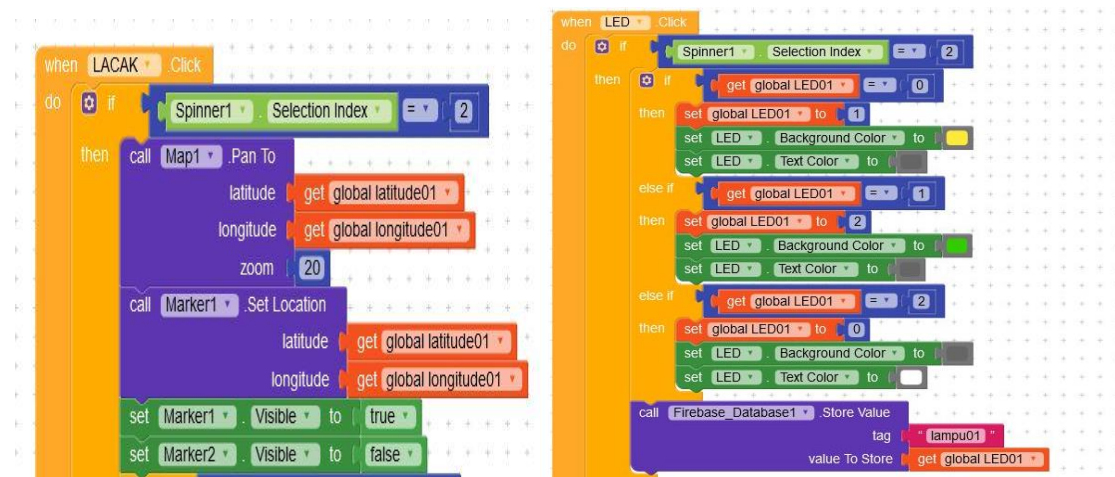
Gambar 41. Frame screen initialize

Setelah firebase berhasil dipanggil, dilanjutkan dengan pemanggilan data menggunakan frame *database got value*. Frame ini hanya berfungsi sekali saat aplikasi baru bekerja. Fungsinya adalah untuk memanggil data yang sudah ada pada firebase. Selanjutnya adalah frame *database data changed*, frame ini berfungsi untuk memanggil data setiap kali *database* menerima data baru. Gambar 42 menunjukkan frame *database got value* dan *database data change*.



Gambar 42. Frame firebase got value dan data changed

Selanjutnya untuk melakukan perintah lacak dan pengaktifan *LED* dilakukan dengan menggunakan frame button *click*. Frame ini berfungsi untuk mengeksekusi *block-block* program yang ditempatkan pada frame ini apabila tombol yang dimaksud ditekan. Gambar 43 menunjukkan frame button *LACAK* click dan frame button *LED* click.



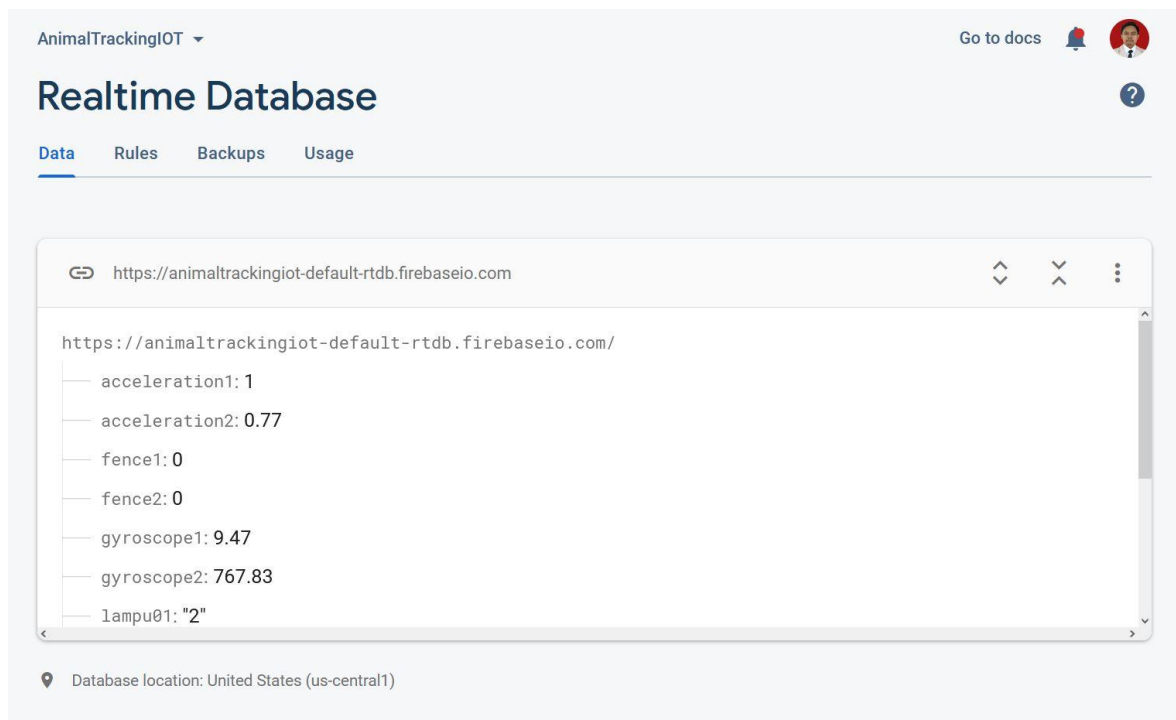
Gambar 43. Frame button LACAK dan button LED

4.4.3. Database

Aplikasi ini menggunakan dua jenis *database*. Yang pertama adalah *real-time database* yang disediakan oleh firebase, dan yang kedua adalah *storage database* yang disediakan oleh airtable.

Real-time database berfungsi sebagai penampung data sementara. Data dari alat *animal tracking* akan dikirim ke *real-time database* kemudian data tersebut akan dipanggil oleh

aplikasi. *Real-time* database yang disediakan oleh firebase ini akan menyimpan data yang dikirimkan oleh alat *animal tracking* pada suatu variable. Apabila alat *animal tracking* mengirim lagi data dengan variable yang sama maka data yang sebelumnya disimpan di *real-time database* ini akan tergantikan. Hal ini akan memudahkan aplikasi dalam menampilkan data karena hanya ada satu data untuk satu variable. Gambar 44 menunjukkan *real-time database* yang digunakan pada penelitian ini.



Gambar 44. *Firestore real-time database*

Storage *database* berfungsi untuk menyimpan data yang sudah ataupun belum diolah oleh aplikasi secara permanen dan terurut. Data yang didapat dari alat ataupun yang tersimpan sementara di *real-time database* tidak bisa langsung dikirim ke storage *database* data-data ini harus melalui aplikasi terlebih dahulu. Hal ini dilakukan untuk mencegah adanya data kembar yang dikirim berulang-ulang oleh alat apabila langsung dikirim ke storage *database*. Selain menyortir data kembar, aplikasi disini juga berperan untuk mengolah data sebelum dikirimkan ke storage *database*. Gambar 45 menunjukkan storage *database* yang digunakan dalam penelitian ini.

	latitude_1	longitude_1	latitude_2	longitude_2	gyro_1	gyro_2	accel_1	accel_2	fence_1	fence_2	A waktu
1	-8.54695	115.08445	-8.5469	115.08441	14.1	15.3	3	2	1	0	07/24/2022 04:38 PM
2	-8.54695	115.08445	-8.5469	115.08441	14.1	15.3	3	2	1	0	07/24/2022 04:38 PM
3	-8.54695	115.08445	-8.5469	115.08446	14.1	15.3	3	2	1	0	07/24/2022 04:40 PM
4	-8.54695	115.08445	-8.5469	115.08446	14.1	15.3	3	2	1	0	07/24/2022 04:42 PM
5	-8.54695	115.08445	-8.5469	115.08446	14.4	15.3	3	2	1	0	07/24/2022 04:42 PM
6	-8.54695	115.08445	-8.54692	115.08446	14.4	15.3	3	2	1	0	07/24/2022 04:43 PM
7	-8.54695	115.08445	-8.54692	115.08446	14.4	15.3	3	2	1	0	07/24/2022 04:44 PM
8	-8.54695	115.08445	-8.54692	115.08446	14.4	15.3	3	2	1	0	07/24/2022 04:46 PM
9	-8.54695	115.08445	-8.54692	115.08446	14.4	15.3	3	3	1	0	07/24/2022 04:46 PM
10	-8.54695	115.08445	-8.54692	115.08446	14.4	15.3	4	3	1	0	07/24/2022 04:46 PM
11	-8.54695	115.08445	-8.54692	115.08446	14.4	15.3	4	2	1	0	07/24/2022 04:48 PM
12	-8.54695	115.08445	-8.54692	115.08446	14.4	15.3	7	2	1	0	07/24/2022 04:49 PM
13	-8.54695	115.08445	-8.54692	115.08446	14.4	15.6	7	2	1	0	07/24/2022 04:49 PM
14	-8.54695	115.08445	-8.54692	115.08446	14.4	15.6	7	2	1	0	07/24/2022 04:50 PM
15	-8.54695	115.08445	-8.54692	115.08446	14.4	15.6	7	2	1	0	07/24/2022 04:53 PM
16	-8.54695	115.08445	-8.54692	115.08446	14.4	15.6	7	3	1	0	07/24/2022 04:53 PM
17	-8.54695	115.08445	-8.54692	115.08446	14.4	15.6	7	3	1	0	07/24/2022 04:53 PM
18	-8.54695	115.08445	-8.54692	115.08446	14.4	15.6	7	3	1	0	07/24/2022 04:55 PM
19	-8.54695	115.08445	-8.54692	115.08446	14.4	15.6	7	3	1	0	07/24/2022 04:56 PM
20	-8.54695	115.08445	-8.54692	115.08446	14.4	15.6	7	3	1	0	07/24/2022 04:57 PM
21	-8.54695	115.08445	-8.54692	115.08446	14.4	15.6	7	3	1	0	07/24/2022 04:59 PM
22	-8.54695	115.08445	-8.54692	115.08446	14.4	15.8	7	3	1	0	07/24/2022 05:00 PM
+	-8.54695	115.08446	-8.54692	115.08446	14.4	15.8	7	3	1	0	07/24/2022 05:01 PM

28 records

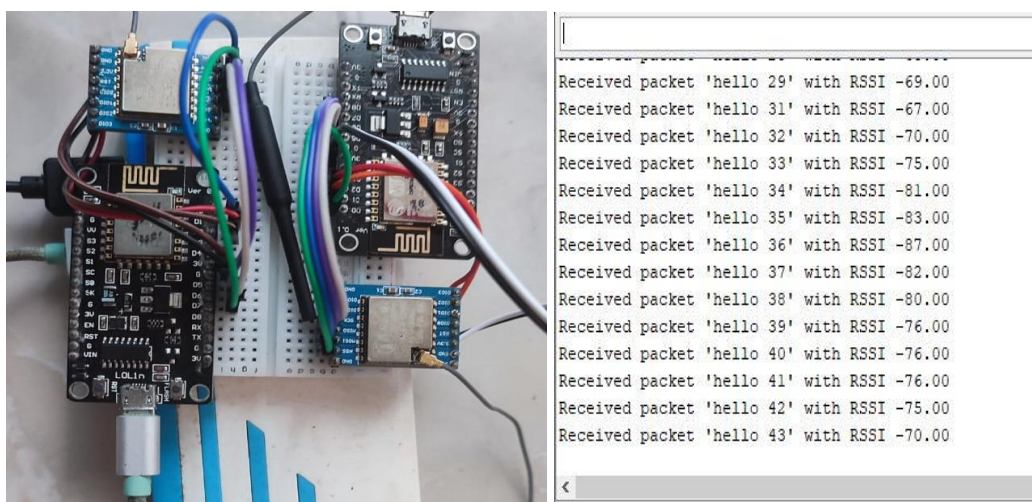
Gambar 45. Airtable data storage

4.2. Pengujian Sistem

4.2.1. Pengujian Hardware

a. Pengujian LoRa

Pengujian *LoRa* dilakukan dengan membangun komunikasi antara dua buah *LoRa*. Sebuah pesan akan dikirimkan setiap detik. Tiap pesan berisi nomer counter sesuai dengan urutan pengiriman serta nilai *RSSI* untuk tiap kali pengiriman. Pengujian dilakukan untuk memastikan bahwa *LoRa* dapat bekerja dengan baik. Gambar 46 menunjukkan pengujian kerja *LoRa*.

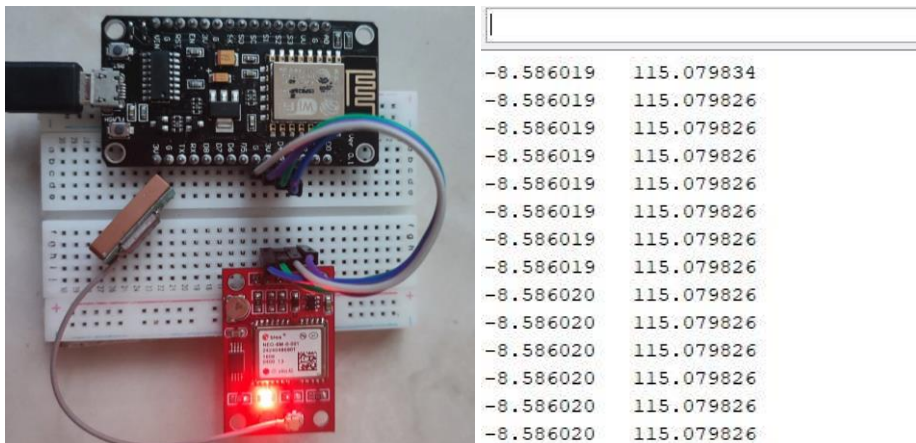


Gambar 46. Pengujian kerja LoRa

Dari gambar dapat dilihat bahwa *LoRa* dapat saling berkomunikasi tanpa adanya kehilangan data. Hal ini menunjukkan *LoRa* dapat berfungsi dengan baik.

b. Pengujian Modul *GPS*

Pengujian *GPS* dilakukan dengan menyalakan modul *GPS* dan menunggu hingga modul *GPS* merespon. Pengujian ini bertujuan untuk memastikan apakah modul *GPS* mampu bekerja dengan baik. Gambar 47 merupakan pengujian *GPS*.

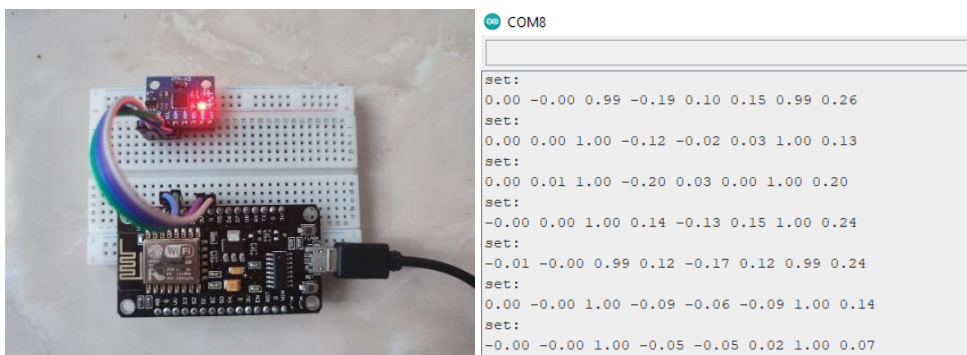


Gambar 47. pengujian *GPS*

Dari gambar terlihat bahwa modul *GPS* mampu menerima dan menampilkan nilai *latitude* dan *longitude* dari lokasi pengujian. Hal ini berarti modul *GPS* mampu bekerja dengan baik.

c. Pengujian Sensor *MPU6050*

Pengujian Sensor *MPU6050* dilakukan dengan menyalakan sensor dan melihat respon sensor terhadap gerakan. Gambar 48 menunjukkan pengujian sensor *MPU6050*



Gambar 48. pengujian sensor *MPU6050*

Dari Gambar 48, terlihat respon dari sensor *MPU6050*. Hal ini menunjukkan bahwa sensor sudah bekerja dengan baik.

4.2.2. Pengujian Program & Aplikasi

a. Pengujian Kerja Program

Pengujian program dilakukan dengan pengimplementasian program yang sudah dibuat secara langsung pada perangkat *master* dan *client*. Pengujian ini bertujuan untuk mengetahui apakah program yang dibuat sudah mampu menjalankan tugasnya dengan baik. Gambar 49 menunjukkan serial monitor perangkat *master* saat pengujian kerja program.

```
COM5
Data Dari Node 1: 0.000000 0.000000
1.57 <-> 1.05
Geo_Fence: 0
Data Dari Node 2: -8.586029 115.079790
0.78 <-> 767.78
Geo_Fence: 0
Data Dari Node 1: 0.000000 0.000000
1.57 <-> 1.14
Geo_Fence: 0
Data Dari Node 2: -8.586021 115.079770
0.78 <-> 768.57
Geo_Fence: 0
Data Dari Node 1: 0.000000 0.000000
```

Gambar 49. Serial monitor perangkat *master* saat pengujian kerja program

Dari gambar bisa dilihat bahwa perangkat *master* berhasil menerima data *latitude*, *longitude*, *accelerasi*, *gyro* dan *geo-fence* dari kedua *client*. Hal ini menunjukkan bahwa program yang dibuat mampu menjalankan tugasnya dengan baik.

b. Pengujian Kerja Aplikasi

Fungsi aplikasi dalam penelitian ini adalah sebagai media monitoring dan remote control untuk menyalakan *LED* pada perangkat *client*. Gambar 50 menunjukkan aplikasi mampu menampilkan nilai *latitude*, *longitude*, keadaan *geo-fence* dan keadaan hewan ternak serta mampu menampilkan koordinat pada maps.



Gambar 50. Tampilan data pada aplikasi

Dari gambar terlihat bahwa aplikasi mampu menampilkan data terkini yang didapat dari alat *animal tracking*.

4.3. Pembahasan

Setelah alat dan aplikasi selesai dibuat, dilanjutkan dengan beberapa pengujian yang bertujuan untuk mengetahui kemampuan sistem ini dalam menjalankan tugasnya. Adapun hasil yang didapat adalah sebagai berikut.

4.3.1. Data Hasil Pengujian Kemampuan Lacak Lokasi

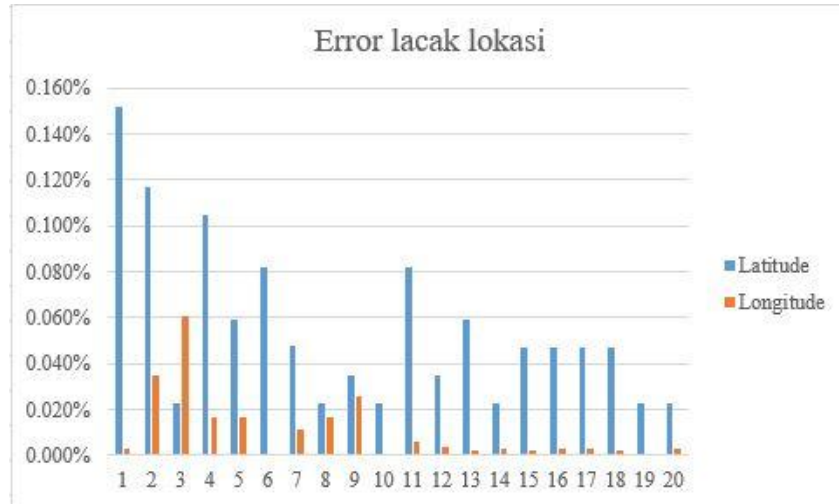
Pengujian kemampuan lacak lokasi dilakukan sebanyak 20 kali di dalam area peternakan. Adapun hasil yang didapat adalah seperti yang ditunjukkan oleh tabel 2 data hasil pengujian kemampuan lacak lokasi.

Tabel 2. Data hasil pengujian kemampuan lacak lokasi

Pengujian ke-	Sampel		Acuan		Presentase Selisih	
	<i>Latitude</i>	<i>Longitude</i>	<i>Latitude</i>	<i>Longitude</i>	<i>Latitude</i>	<i>Longitude</i>
1	-8.54695	115.0845	-8.54682	115.0845	0.152%	0.003%
2	-8.5468	115.0846	-8.5469	115.085	0.117%	0.035%
3	-8.54682	115.0845	-8.54684	115.0838	0.023%	0.061%
4	-8.54693	115.0839	-8.54684	115.0841	0.105%	0.017%
5	-8.54645	115.0844	-8.5465	115.0846	0.059%	0.017%
6	-8.54646	115.0845	-8.54653	115.0845	0.082%	0.000%
7	-8.5463	115.0844	-8.54634	115.0845	0.048%	0.011%
8	-8.54645	115.0845	-8.54647	115.0847	0.023%	0.017%
9	-8.54659	115.085	-8.54662	115.0853	0.035%	0.026%
10	-8.54634	115.0845	-8.54632	115.0845	0.023%	0.001%
11	-8.54686	115.0847	-8.54679	115.0848	0.082%	0.006%
12	-8.54685	115.0846	-8.54688	115.0846	0.035%	0.004%
13	-8.54691	115.0847	-8.54686	115.0847	0.059%	0.002%
14	-8.54694	115.0846	-8.54696	115.0845	0.023%	0.003%
15	-8.54681	115.0849	-8.54677	115.0848	0.047%	0.002%
16	-8.54679	115.0848	-8.54683	115.0849	0.047%	0.003%
17	-8.54697	115.0847	-8.54701	115.0847	0.047%	0.003%
18	-8.54699	115.0848	-8.54703	115.0848	0.047%	0.002%
19	-8.54672	115.0845	-8.54674	115.0845	0.023%	0.001%
20	-8.54668	115.0846	-8.5467	115.0846	0.023%	0.003%
Rata-rata presentase selisih					0.055%	0.011%

Tabel 2 menunjukkan nilai *latitude* dan *longitude* dari lokasi yang sama namun menggunakan perangkat yang berbeda. Sampel merupakan nilai *latitude* dan *longitude* yang didapatkan dari alat *animal tracking*, sedangkan acuan adalah nilai yang didapat dari Google Maps menggunakan *GPS* bawaan dari smartphone. Berdasarkan hasil yang didapat, alat

animal tracking mampu memetakan lokasi suatu objek dengan akurat. Akurasi ini terlihat dari hasil persentase selisih yang sangat kecil bahkan tidak sampai 1% untuk tiap pengujian dan tidak menyentuh 0,1% jika dirata-ratakan. Gambar 51 menunjukkan grafik presentase error lacak lokasi.



Gambar 51. Grafik presentase error lacak lokasi

4.3.2. Data Hasil Pengujian Kemampuan Memindai Gerakan

Pengujian kemampuan memindai gerakan dilakukan sebanyak 10 kali untuk tiap tipe gerakan. Gerakan terjatuh dilakukan dengan simulasi, sedangkan maka gerakan berbaring dan berdiri dilakukan secara langsung pada hewan ternak. Tabel 3 dan 4 menampilkan data hasil pengujian kemampuan memindai gerakan. Nilai 1 pada kolom respon berarti sistem mampu dengan tepat mengidentifikasi suatu gerakan. Nilai 0 pada kolom respon berarti sistem gagal dalam mengidentifikasi suatu gerakan baik itu gagal sepenuhnya ataupun salah dalam mendefinisikan gerakan. Kolom delay menunjukkan lama sistem dalam menentukan suatu gerakan dimulai dari saat gerakan terjadi. Hasil delay yang didapat tetap dicatat meskipun hasil pembacaan gerakan tidak tepat.

Tabel 3. Data hasil memindai gerakan Terjatuh ke depan, belakang dan kanan

Pengujian ke-	Terjatuh ke depan		Terjatuh ke belakang		Terjatuh ke kanan	
	Respon	Delay(S)	Respon	Delay(S)	Respon	Delay(S)
1	0	4	1	1	1	1
2	0	4	0	4	0	4
3	1	1	0	5	0	5
4	1	2	0	3	1	2
5	0	5	1	2	1	1

6	0	1	0	6	0	7
7	0	5	1	2	1	1
8	1	1	1	1	1	3
9	0	3	1	1	0	6
10	1	1	0	7	1	1

Tabel 4. Data hasil memindai gerakan Terjatuh ke kiri, gerakan makan dan berdiri

Pengujian ke-	Terjatuh ke kiri		Berbaring		Berdiri	
	Respon	Delay(S)	Respon	Delay(S)	Respon	Delay(S)
1	0	7	1	6	1	5
2	1	1	0	2	1	2
3	1	2	1	3	1	2
4	0	5	1	2	1	2
5	0	5	1	4	1	2
6	1	2	0	1	1	2
7	1	2	1	5	1	2
8	0	7	1	5	1	2
9	1	1	0	5	1	2
10	1	1	1	3	1	2

Dari tabel 3 dan 4, dapat dicari presentase keberhasilan dan juga rata-rata delay dari kemampuan sistem dalam memindai gerakan tiap gerakan. Tabel 5 menunjukkan presentase keberhasilan dan rata-rata delay dari tiap gerakan yang diujikan.

Tabel 5. Presentase keberhasilan dan delay pengujian memindai gerakan

Pengujian Gerakan	% Keberhasilan	Delay(S)
Terjatuh ke depan	40%	2.444444
Terjatuh ke belakang	50%	3.2
Terjatuh ke kanan	60%	3.1
Terjatuh ke kiri	60%	3.3
Berbaring	70%	3.6
Berdiri	100%	2.3

Berdasarkan hasil yang didapat, gerakan terjatuh memiliki tingkat keberhasilan yang cukup rendah dikisaran 40% hingga 60% saja. Hal ini terjadi karena sistem salah mendefinisikan arah terjatuh. Namun apabila arah terjatuh diabaikan, sistem ini mampu

mendefinisikan gerakan terjatuh dengan tingkat keberhasilan 80% dengan delay 2,7 detik.

Tabel 6 menunjukkan hasil memindai gerakan terjatuh.

Tabel 6. Hasil memindai gerakan terjatuh

Pengujian ke-	Respon	Delay(S)
1	1	2
2	1	1
3	0	3
4	1	2
5	1	2
6	1	1
7	1	4
8	0	5
9	1	4
10	1	3
% Keberhasilan	80	
Delay rata-rata		2.7

4.3.3. Data Hasil Pengujian Keandalan *Geo-fencing*

Pengujian *geo-fencing* dibagi menjadi dua, yaitu *geo-fencing* sebagai pembatas area peternakan dan *geo-fencing* sebagai pelacak pergerakan temporal hewan ternak. Pengujian sebagai pembatas area peternakan dilakukan dengan memindahkan perangkat *client* keluar dari area peternakan. Untuk pengujian pelacak gerakan temporal dilakukan dengan memindahkan perangkat *client* masuk kedalam area-area yang sudah dibuat. Kedua pengujian ini dilakukan dengan simulasi. Tabel 7 menunjukkan jarak error *geo-fencing* pembatas area peternakan dan akurasi sistem dalam melacak pergerakan temporal hewan ternak.

Tabel 7. Data hasil pengujian *geo-fencing*

Pengujian ke-	<i>Geo-fencing</i>	Akurasi area gerakan temporal	
	Jarak eror(m)	Istirahat	Merumput
1	4	1	1
2	5	1	1
3	4	1	1
4	2	0	1

5	3	1	1
6	5	1	0
7	7	1	1
8	3	1	1
9	3	1	1
10	8	1	0
Rata-rata	4.4		
% keberhasilan		90	80

Pada *geo-fencing* sebagai pembatas area peternakan didapat hasil yang bervariasi mulai dari yang terkecil 2 meter pada pengujian ke-4 hingga 8 meter pada pengujian ke-10. Namun apabila dirata-ratakan, didapat hasil 4,4 meter yang berarti sistem ini mampu bekerja dengan cukup baik dan akurat. Untuk pergerakan temporal didapat hasil yang cukup baik dengan persentase keberhasilan mencapai 90% di area istirahat dan 80% di area merumput.

4.3.4. Data Hasil Pengujian Pengaruh Jarak Terhadap Kekuatan Sinyal *LoRa*

Pengujian pengaruh jarak terhadap kekuatan sinyal *LoRa* dilakukan sebanyak 10 kali dengan kelipatan jarak 10 meter setiap pengujian. Tabel 8 menunjukkan hasil pengujian pengaruh jarak terhadap kekuatan sinyal *LoRa*.

Tabel 8. Data hasil pengujian pengaruh jarak terhadap kekuatan sinyal LoRa

Pengujian ke-	Jarak(m)	Nilai <i>RSSI</i> (<i>dBm</i>)
1	10	-64
2	20	-78
3	30	-89
4	40	-99
5	50	-102
6	60	-111
7	70	-118
8	80	-122
9	90	-124
10	100	-131

Dari tabel diatas, dapat dilihat bahwa nilai *RSSI* berbanding terbalik dengan jarak. Semakin jauh jarak maka semakin kecil pula nilai *RSSI* yang didapat. Gambar 52 menunjukkan grafik pengaruh jarak terhadap nilai *RSSI*.



Gambar 52. Pengujian RSSI

BAB V

PENUTUP

5.1. Kesimpulan

Penelitian ini telah berhasil memaparkan implementasi *Internet of things (Iot)*, *movement recognition* dan *geo-fencing* dengan *Global Positioning Sistem (GPS)* yang terintegrasi dengan teknologi *Long Range (LoRa)* guna memonitoring keberadaan dan keadaan hewan ternak. Adapun beberapa kesimpulan yang didapat adalah sebagai berikut:

- a. Dengan mengimplementasikan *IOT* dapat dilakukan monitoring keberadaan hewan ternak. Penelitian ini berhasil mengimplementasikan sistem *IOT* dengan menggunakan *GPS* yang mampu melacak lokasi ternak dengan tingkat akurasi yang sangat tinggi. Saat dibandingkan dengan Google maps, didapat error yang sangat kecil, yaitu 0,055% pada *latitude* dan 0,011% pada *longitude*. Sedangkan untuk penyimpanan data menggunakan *real-time database* dari Firebase dan untuk data logger menggunakan *Airtable*
- b. Pemindaian gerakan hewan ternak bisa dilakukan dengan pengimplementasian sistem *movement recognition* menggunakan sensor *gyroscope* dan *accelerometer*. Sistem *movement recognition* pada penelitian ini mampu memindai tiga jenis gerakan hewan ternak dengan cukup baik. Pemindaian gerakan berbaring memiliki tingkat keberhasilan sebesar 70%, gerakan berdiri memiliki tingkat keberhasilan 100%. Sedangkan untuk gerakan terjatuh memiliki keberhasilan sebesar 80%.
- c. Dengan mengimplementasikan sistem *GPS* yang memetakan bumi dengan sumbu *latitude* dan *longitude*. Dapat dibuat batas-batas tertentu yang dapat didefinisikan sebagai pagar virtual. Batas-batas tersebut dapat dibuat menggunakan konsep sudut bangun datar yang dipadukan dengan konsep perpindahan dua buah vektor. Penelitian ini berhasil mengimplementasikan sistem *GPS* guna membangun sistem *geo-fencing*. Sistem *geo-fencing* yang sudah dibuat memiliki rata-rata jarak error sebesar 4,4 meter. untuk pembacaan area istirahat mempunyai tingkat keberhasilan 90%, sedangkan area merumput memiliki keberhasilan sebesar 80%.
- d. Jarak berperan cukup besar terhadap kekuatan sinyal *LoRa*. Dalam penelitian ini didapat nilai *RSSI* sebesar -64dBm pada jarak 10 meter. nilai ini terus mengecil hingga -131dBm pada jarak 100 meter. Hal ini menunjukkan bahwa pengaruh jarak terhadap nilai *RSSI* adalah berbanding terbalik. Semakin jauh jarak maka semakin kecil pula nilai *RSSI*.

5.2. Saran

Berdasarkan hasil yang didapat, maka ada beberapa hal yang perlu dikembangkan lagi yaitu:

- a. Perlu ditambahkan analisa penggunaan power dan pengaruh panel surya terhadap durasi masa pakai baterai.
- b. Perlu dilakukan inovasi untuk dapat menampilkan tampilan map dengan cepat, karena untuk memuat gambar map yang disediakan oleh beberapa penyedia layanan masih terlalu berat.
- c. Perlu ditambahkan suatu fitur yang mampu membedakan nyala LED alat dengan nyala lampu lain.

DAFTAR PUSTAKA

-
- [1] R. Angriawan and N. Anugraha, "Sistem Pelacak Lokasi Sapi dengan Sistem Komunikasi *LoRa*," *Inspiration: Jurnal Teknologi Informasi dan Komunikasi*, vol. 9, no. 1, p. 33, Jun. 2019, doi: 10.35585/inspir.v9i1.2494.
- [2] D. B. Lasfeto, T. Setyorini, and Y. A. A. Lada, "Desain Sistem Monitoring Ternak Sapi Berbasis Jaringan Sensor Nirkabel Untuk Sistem Penggembalaan Lepas Di Timor Barat Provinsi Nusa Tenggara Timur," *Pros. Semin. Nas. Sains dan Teknol. 2017*, vol. 07, no. November, 2017.
- [3] J. Priono and E. B. Setiawan, "Implementasi Geofencing dalam Mengawasi Pengiriman Kendaraan di Sebuah Perusahaan Ekspedisi," *106 Ultim.*, vol. IX, no. 2, 2017.
- [4] L. Tambunan and D. Putra, "SISTEM KONTROL KENDARAAN BERBASIS *IOT*," 2019. [Online]. Available: <http://ojsamik.amikmitragama.ac.id>
- [5] M. Jimmy *et al.*, "Buku Proceeding | Seminar Nasional Efisiensi Energi untuk Peningkatan Daya Saing Industri Manufaktur & Otomotif Nasional (SNEEMO) Jakarta," 2020.
- [6] D. C. Mahendra, T. Susyanto, and S. Siswanti, "SISTEM MONITORING MOBIL RENTAL MENGGUNAKAN *GPS* TRACKER," *Jurnal Ilmiah SINUS*, vol. 16, no. 2, Aug. 2018, doi: 10.30646/sinus.v16i2.357.
- [7] U. Suwardoyo and M. J. Jayadi, "MENGGUNAKAN *GPS* (*GLOBAL POSITION SYSTEM*) BERBASIS WEB," vol. 1, no. 3, 2021, doi: 10.31850/jsilog.v1i3.
- [8] M. Fadhiil and F. S. Jurusan, "Rancang Bangun Emergency Button Berbasis *LORA*," 2020.
- [9] Y. Triwidyastuti, "Performance Analysis of Point-to-Point *LoRa* End Device Communication," *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, vol. 10, no. 3, p. 140, Dec. 2019, doi: 10.24843/lkjiti.2019.v10.i03.p02.
- [10] S. S. A. Yanzhiah, "edhy-sst-journal-manager-technoscintia-vol-13-no-02-09-hal-059-067-asma-yanzhiah-analisis-jarak-jangkauan," *Anal. JARAK JANGKAUAN LORA DENGAN Param. RSSI DAN Pack. LOSS PADA AREA URBAN*, vol. 13, no. 1, pp. 59–67, 2020.
- [11] F. Akrom Zulhij Fajri and M. S. Mauludin, "Rancang Bangun Sistem Keamanan Aliran Listrik Arus AC dengan Fingerprint menggunakan Arduino Nano," *J. Inform. dan Rekayasa Perangkat Lunak*, vol. 2, no. 1, p. 26, 2020, doi: 10.36499/jinrpl.v2i1.3189.
- [12] M. I. Munabbih, E. D. Widiyanto, Y. E. Windarto, and E. Y. Indrasto, "RANCANG BANGUN SISTEM PEMANTAU KUALITAS UDARA MENGGUNAKAN ARDUINO DAN *LORA* BERBASIS JARINGAN SENSOR NIRKABEL," *Transmisi*, vol. 22, no. 1, pp. 6–14, Mar. 2020, doi: 10.14710/transmisi.22.1.6-14.
- [13] E. Didik Widiyanto, A. A. Faizal, D. Eridani, R. Dwi, O. Augustinus, and M. S. Pakpahan, "Simple *LoRa* Protocol: Protokol Komunikasi *LoRa* Untuk Sistem Pemantauan Multisensor Simple *LoRa* Protocol: *LoRa* Communication Protocol for Multisensor Monitoring Systems," *TELKA*, vol. 5, no. 2, pp. 83–92, 2019.
- [14] M. Ridha Fahliwi, "Sistem *Tracking* Position Berdasarkan Titik Koordinat *GPS* Menggunakan Smartphone," *Jurnal Infomedia*, vol. 2, no. 1, 2017.
- [15] R. Setiawan, H. H. Triharminto, and M. Fahrurozi, "Gesture Control Menggunakan IMU MPU 6050 Metode Kalman Filter Sebagai Kendali Quadcopter," *Pros. Semin. Nas. Sains Teknol. dan Inov. Indones.*, vol. 3, no. November, pp. 411–422, 2021, doi:

10.54706/senastindo.v3.2021.133.

- [16] Y. Prabowo, S. Broto, G. P. Utama, G. Gata, and Y. Yuliazmi, “Pengenalan dan Penerapan Pembangkit Listrik Tenaga Surya di Desa Muara Kilis Kabupaten Tebo Jambi,” *Abdimas J. Pengabd. Masy. Univ. Merdeka Malang*, vol. 5, no. 1, pp. 70–78, 2020, doi: 10.26905/abdimas.v5i1.3555.
- [17] R. Hasrul *et al.*, “Analisis Efisiensi Panel Surya Sebagai Energi Alternatif,” vol. 5, no. 9, pp. 79–87, 2021.
- [18] J. Adhyaksa and K. No, “270957-Analisa-Rancangan-Sel-Surya-Dengan-Kapas-505Ef9B9,” *J. Tek. Mesin UNISKA*, vol. 01, no. 02, pp. 33–39, 2016.
- [19] B. H. Purwoto, J. Jatmiko, M. A. Fadilah, and I. F. Huda, “Efisiensi Penggunaan Panel Surya sebagai Sumber Energi Alternatif,” *Emit. J. Tek. Elektro*, vol. 18, no. 1, pp. 10–14, 2018, doi: 10.23917/emit.v18i01.6251.
- [20] R. Baharuddin, “Rancang Bangun Sistem Mini Pembangkit Listrik Tenaga Surya (PLTS) Portable,” *JTT (Jurnal Teknol. Terpadu)*, vol. 9, no. 1, pp. 65–70, 2021, doi: 10.32487/jtt.v9i1.1087.

LAMPIRAN

Lampiran 1. Tabel data hasil pengujian

1	-	8.54695	115.0845	-8.5469	115.0844	14.1	15.3	3	2	1	0	07/24/2022 04:38 PM
2	-	8.54695	115.0845	-8.5469	115.0844	14.1	15.3	3	2	1	0	07/24/2022 04:38 PM
3	-	8.54695	115.0845	-8.5469	115.0845	14.1	15.3	3	2	1	0	07/24/2022 04:40 PM
4	-	8.54695	115.0845	-8.5469	115.0845	14.1	15.3	3	2	1	0	07/24/2022 04:42 PM
5	-	8.54695	115.0845	-8.5469	115.0845	14.4	15.3	3	2	1	0	07/24/2022 04:42 PM
6	-	8.54695	115.0845	8.54692	115.0845	14.4	15.3	3	2	1	0	07/24/2022 04:43 PM
7	-	8.54695	115.0845	8.54692	115.0845	14.4	15.3	3	2	1	0	07/24/2022 04:44 PM
8	-	8.54695	115.0845	8.54692	115.0845	14.4	15.3	3	2	1	0	07/24/2022 04:46 PM
9	-	8.54695	115.0845	8.54692	115.0845	14.4	15.3	3	3	1	0	07/24/2022 04:46 PM
10	-	8.54695	115.0845	8.54692	115.0845	14.4	15.3	4	3	1	0	07/24/2022 04:46 PM
11	-	8.54695	115.0845	8.54692	115.0845	14.4	15.3	4	2	1	0	07/24/2022 04:48 PM
12	-	8.54695	115.0845	8.54692	115.0845	14.4	15.3	7	2	1	0	07/24/2022 04:49 PM
13	-	8.54695	115.0845	8.54692	115.0845	14.4	15.6	7	2	1	0	07/24/2022 04:49 PM
14	-	8.54695	115.0845	8.54692	115.0845	14.4	15.6	7	2	1	0	07/24/2022 04:50 PM
15	-	8.54695	115.0845	8.54692	115.0845	14.4	15.6	7	2	1	0	07/24/2022 04:53 PM
16	-	8.54695	115.0845	8.54692	115.0845	14.4	15.6	7	3	1	0	07/24/2022 04:53 PM
17	-	8.54695	115.0845	8.54692	115.0845	14.4	15.6	7	3	1	0	07/24/2022 04:55 PM
18	-	8.54695	115.0845	8.54692	115.0845	14.4	15.6	7	3	1	0	07/24/2022 04:55 PM
19	-	8.54695	115.0845	8.54692	115.0845	14.4	15.6	7	3	1	0	07/24/2022 04:56 PM
20	-	8.54695	115.0845	8.54692	115.0845	14.4	15.6	7	3	1	0	07/24/2022 04:57 PM

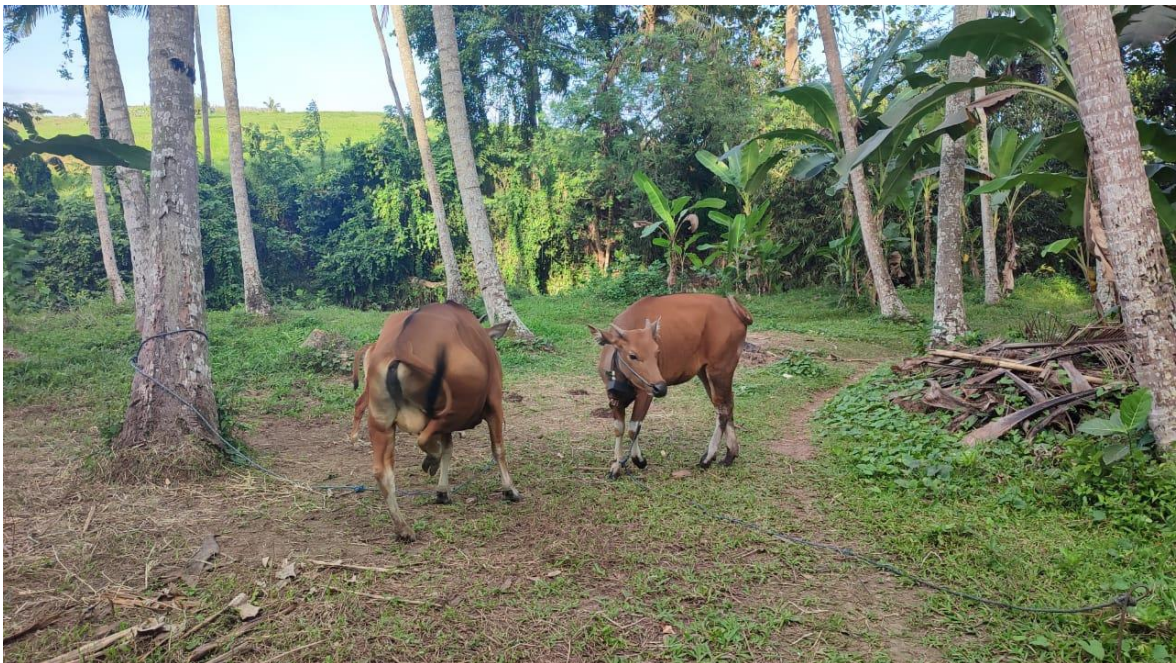
Lampiran 2. Tabel pengenalan gerakan

X	Y	Z	Roll	Pict	Yaw	Accel	gyro
0.01311	0.14736	0.97718	-18.6861	-10.7406	0.12592	0.99323	2.42656
0.01311	0.00331	0.97718	-18.6861	-10.7406	0.12592	0.99323	2.42656
0.01311	0.00331	0.97259	-18.6861	-10.7406	0.12592	0.99323	2.42656
0.01311	0.00331	0.97259	48.38316	-10.7406	0.12592	0.99323	2.42656
0.01311	0.00331	0.97259	48.38316	24.11581	0.12592	0.99323	2.42656
0.01311	0.00331	0.97259	48.38316	24.11581	0.86851	0.99323	2.42656
0.01311	0.00331	0.97259	48.38316	24.11581	0.86851	0.97268	2.42656
-	0.02083	1.08587	4.6427	-6.90709	4.26126	1.0863	9.34991
-	0.91487	1.08587	4.6427	-6.90709	4.26126	1.0863	9.34991
-	0.91487	1.30019	4.6427	-6.90709	4.26126	1.0863	9.34991
-	0.91487	1.30019	0.84246	4.6427	4.26126	1.0863	9.34991
-	0.91487	1.30019	0.84246	12.23049	4.26126	1.0863	9.34991
-	0.91487	1.30019	0.84246	12.23049	-7.57884	4.26126	9.34991
-	0.91487	1.30019	0.84246	12.23049	-7.57884	50.4292	9.34991
-	0.91487	1.30019	0.84246	12.23049	-7.57884	50.4292	1.79923
-	0.91487	1.30019	0.84246	12.23049	-7.57884	50.4292	1.79923
-	0.96663	1.30019	0.84246	12.23049	-7.57884	50.4292	1.79923
-	0.96663	1.33339	0.84246	12.23049	-7.57884	50.4292	1.79923
-	0.96663	1.33339	0.76214	12.23049	-7.57884	50.4292	1.79923
-	0.96663	1.33339	0.76214	15.43659	-7.57884	50.4292	1.79923

Lampiran 3. Area peternakan



Lampiran 4. Tenak yang digunakan sebagai objek uji coba





Lampiran 5. Pengujian jarak eror geo-fencing



```

#include <SPI.h>
#include <LoRa.h>
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <MPU6050_light.h>

MPU6050 mpu(Wire);
#define nss 10
#define rst 9
#define dio0 8
#define lamp 5
#define buzz 2

static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;
TinyGPSPlus gps;
SoftwareSerial ss(RXPin, TXPin);

String outgoing;          // outgoing message
unsigned long timer = 0;
byte msgCount = 0;       // count of outgoing messages
byte MasterNode = 0xFF;
byte Node1 = 0xBB;
float la,lo;
String latitude1,longitude1,ACCEL1,Gyro1;

String Mymessage = "";
int kedip;
int targetStatus;
int fence;
char cumulativeAngle[12];

const double fences[1][4][2] = {{{-8.586095, 115.079757},
                                  {-8.585964, 115.079884},
                                  {-8.585900, 115.079769},
                                  {-8.585960, 115.079669},}
};

void setup() {
  Serial.begin(115200);

  Wire.begin();

  pinMode(lamp, OUTPUT);
  pinMode(buzz, OUTPUT);

```

```

while (!Serial);

Serial.println("LoRa Node1");

LoRa.setPins(nss, rst, dio0);

if (!LoRa.begin(433E6)) {
  Serial.println("Starting LoRa failed!");
  while (1);
}
Serial.println("LoRa Online");
ss.begin(GPSBaud);

byte status = mpu.begin();
Serial.print(F("MPU6050 status: "));
Serial.println(status);
while(status!=0){ }

Serial.println(F("Calculating offsets, do not move MPU6050"));
delay(1000);
mpu.calcOffsets(true,true);
Serial.println("Done!\n");
}

void loop() {

mpu.update();
if(millis() - timer > 1000)
{
  float ax = (mpu.getAccX());
  float ay = (mpu.getAccY());
  float az = (mpu.getAccZ());
  float gx = (mpu.getGyroX());
  float gy = (mpu.getGyroY());
  float gz = (mpu.getGyroZ());
  timer = millis();
  ACCEL1 = String(sqrt((ax*ax)+(ay*ay)+(az*az)));
  Gyro1 = String(sqrt((gx*gx)+(gy*gy)+(gz*gz)));
}

while (ss.available() > 0)
  if (gps.encode(ss.read()))
  {
    la = (gps.location.lat());
    lo = (gps.location.lng());
  }
}

```

```

    latitude1 = String(la,6);
    longitude1 = String(lo,6);
}
onReceive(LoRa.parsePacket());
}

void pip(){
    int fenceSize = sizeof(fences[fence - 1])/sizeof(fences[fence - 1][0]);
    double vectors[fenceSize][2];
    for(int i = 0; i < fenceSize; i++){
        vectors[i][0] = fences[fence - 1][i][0] - la;
        vectors[i][1] = fences[fence - 1][i][1] - lo;
    }
    double angle = 0;
    double num, den;
    for(int i = 0; i < fenceSize; i++){
        num = (vectors[i% fenceSize][0])*(vectors[(i+1)% fenceSize][0])+
(vectors[i% fenceSize][1])*(vectors[(i+1)% fenceSize][1]);
        den = (sqrt(pow(vectors[i% fenceSize][0],2) +
pow(vectors[i% fenceSize][1],2)))*(sqrt(pow(vectors[(i+1)% fenceSize][0],2) +
pow(vectors[(i+1)% fenceSize][1],2)));
        angle = angle + (180*acos(num/den)/M_PI);
    }
    dtostrf(angle,9,7,cumulativeAngle);
    if(angle > 355 && angle < 365)
        targetStatus = 1;
    else
        targetStatus = 0;
}

void onReceive(int packetSize) {
    if (packetSize == 0) return;    // if there's no packet, return

    // read packet header bytes:
    int recipient = LoRa.read();    // recipient address
    byte sender = LoRa.read();    // sender address
    byte incomingMsgId = LoRa.read(); // incoming msg ID
    byte incomingLength = LoRa.read(); // incoming msg length

    String incoming = "";

    while (LoRa.available()) {
        incoming += (char)LoRa.read();
    }

    if (incomingLength != incoming.length()) { // check length for error

```

```

// Serial.println("error: message length does not match length");
;
return;          // skip rest of function
}

// if the recipient isn't this device or broadcast,
if (recipient != Node1 && recipient != MasterNode) {
  //Serial.println("This message is not for me.");
  ;
  return;          // skip rest of function
}
Serial.println(incoming);
int Val = incoming.toInt();
if (Val == 10 && targetStatus == 0)
{
  Mymessage = Mymessage +
latitude1+", "+longitude1+", "+ACCEL1+", "+Gyro1+", "+targetStatus;
  sendMessage(Mymessage, MasterNode, Node1);
  delay(100);
  Mymessage = "";
  digitalWrite(lamp, LOW);
  digitalWrite(buzz, HIGH);
}

else if (Val == 10 && targetStatus == 1)
{
  Mymessage = Mymessage +
latitude1+", "+longitude1+", "+ACCEL1+", "+Gyro1+", "+targetStatus;
  sendMessage(Mymessage, MasterNode, Node1);
  delay(100);
  Mymessage = "";
  digitalWrite(lamp, LOW);
  digitalWrite(buzz, LOW);
}
else if (Val == 11 && targetStatus == 0)
{
  Mymessage = Mymessage +
latitude1+", "+longitude1+", "+ACCEL1+", "+Gyro1+", "+targetStatus;
  sendMessage(Mymessage, MasterNode, Node1);
  delay(100);
  Mymessage = "";
  digitalWrite(lamp, HIGH);
  digitalWrite(buzz, HIGH);
}

else if (Val == 11 && targetStatus == 1)

```

```

{
  Mymessage = Mymessage +
latitude1+", "+longitude1+", "+ACCEL1+", "+Gyro1+", "+targetStatus;
  sendMessage(Mymessage, MasterNode, Node1);
  delay(100);
  Mymessage = "";
  digitalWrite(lamp, HIGH);
  digitalWrite(buzz, LOW);
}
else if (Val == 12)
{
  Mymessage = Mymessage +
latitude1+", "+longitude1+", "+ACCEL1+", "+Gyro1+", "+targetStatus;
  sendMessage(Mymessage, MasterNode, Node1);
  delay(100);
  Mymessage = "";
  digitalWrite(lamp, HIGH);
  digitalWrite(buzz, LOW);
}

else if (Val == 13)
{
  Mymessage = Mymessage +
latitude1+", "+longitude1+", "+ACCEL1+", "+Gyro1+", "+targetStatus;
  sendMessage(Mymessage, MasterNode, Node1);
  delay(100);
  Mymessage = "";
  digitalWrite(lamp, LOW);
  digitalWrite(buzz, LOW);
}
}

void sendMessage(String outgoing, byte MasterNode, byte Node1) {
  LoRa.beginPacket();          // start packet
  LoRa.write(MasterNode);      // add destination address
  LoRa.write(Node1);          // add sender address
  LoRa.write(msgCount);       // add message ID
  LoRa.write(outgoing.length()); // add payload length
  LoRa.print(outgoing);       // add payload
  LoRa.endPacket();           // finish packet and send it
  msgCount++;                 // increment message ID
}

```

```

#include <SPI.h>
#include <LoRa.h>
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>

#define FIREBASE_HOST "animaltrackingiot-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH
"j3iIVZ5PBH6mGGlnJpBAbb8Uvi7rzWhwis4H3Lxt"
#define WIFI_SSID "JohnSantuy"
#define WIFI_PASSWORD "Guardian123"
#define nss D0
#define rst D1
#define dio0 D2

byte MasterNode = 0xFF;
byte Node1 = 0xBB;
byte Node2 = 0xCC;

String SenderNode = "";
String outgoing;          // outgoing message

byte msgCount = 0;       // count of outgoing messages
String incoming = "";

// Tracks the time since last event fired
unsigned long previousMillis = 0;
unsigned long int previoussecs = 0;
unsigned long int currentsecs = 0;
unsigned long currentMillis = 0;
int interval = 1 ; // updated every 1 second
int Secs = 0;

float
latitude1,longitude1,latitude2,longitude2,acceleration1,gyroscope1,acceleration2,
gyroscope2;
int GFence1,GFence2;
int lamp1 = 0;
int lamp2 = 0;
int buzz = 0;

void setup() {
  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {

```



```

Serial.print(".");
delay(500);
Serial.println("");
Serial.print("connected: ");
Serial.println(WiFi.localIP());
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}
Serial.println("LoRa Master Node");
LoRa.setPins(nss, rst, dio0);
if (!LoRa.begin(433E6)) {
  Serial.println("Starting LoRa failed!");
  while (1);
}
Serial.print("LoRa Online");
}

void loop() {

  String lampu1 = Firebase.getString("lampu01");
  String lampu2 = Firebase.getString("lampu02");
  String buzzer = Firebase.getString("buzz");
  lamp1 = lampu1.toInt();
  lamp2 = lampu2.toInt();
  buzz = buzzer.toInt();
  currentMillis = millis();
  currentsecs = currentMillis / 1000;

  if ((unsigned long)(currentsecs - previoussecs) >= interval) {
    Secs = Secs + 1;
    //Serial.println(Secs);
    if ( Secs >= 4 )
    {
      Secs = 0;
    }

    if ( (Secs >= 0) && (Secs < 1) && (lamp1 == 0) && (buzz == 0))
    {
      String message = "10";
      sendMessage(message,MasterNode, Node1);
      //Serial.println("10");
    }

    else if ( (Secs >= 0) && (Secs < 1) && (lamp1 == 1) && (buzz == 0))
    {
      String message = "11";
      sendMessage(message,MasterNode, Node1);
    }
  }
}

```

```

}

if ( (Secs >= 0) && (Secs < 1) && (lamp1 == 0) && (buzz == 1))
{
String message = "13";
sendMessage(message,MasterNode, Node1);
//Serial.println("10");
}

else if ( (Secs >= 0) && (Secs < 1) && (lamp1 == 1) && (buzz == 1))
{
String message = "12";
sendMessage(message,MasterNode, Node1);
}

if ( (Secs >= 2 ) && (Secs < 3) && (lamp2 == 0) && (buzz == 0))
{
String message = "20";
sendMessage(message,MasterNode, Node2);
//Serial.println("20");
}

else if ( (Secs >= 2 ) && (Secs < 3) && (lamp2 == 1) && (buzz == 0))
{
String message = "21";
sendMessage(message,MasterNode, Node2);
//Serial.println("21");
}

if ( (Secs >= 2 ) && (Secs < 3) && (lamp2 == 0) && (buzz == 1))
{
String message = "23";
sendMessage(message,MasterNode, Node2);
//Serial.println("20");
}

else if ( (Secs >= 2 ) && (Secs < 3) && (lamp2 == 1) && (buzz == 1))
{
String message = "22";
sendMessage(message,MasterNode, Node2);
//Serial.println("21");
}

previoussecs = currentsecs;
}

```

```
    onReceive(LoRa.parsePacket());  
}
```

```
void sendMessage(String outgoing, byte MasterNode, byte otherNode) {  
    LoRa.beginPacket();           // start packet  
    LoRa.write(otherNode);        // add destination address  
    LoRa.write(MasterNode);       // add sender address  
    LoRa.write(msgCount);         // add message ID  
    LoRa.write(outgoing.length()); // add payload length  
    LoRa.print(outgoing);         // add payload  
    LoRa.endPacket();             // finish packet and send it  
    msgCount++;                   // increment message ID  
    //Serial.println("Paket Terkirim");  
}
```

```
void onReceive(int packetSize) {  
    if (packetSize == 0) return;   // if there's no packet, return  
  
    // read packet header bytes:  
    int recipient = LoRa.read();    // recipient address  
    byte sender = LoRa.read();     // sender address  
    if ( sender == 0XBB )  
        SenderNode = "Node1:";  
    if ( sender == 0XCC )  
        SenderNode = "Node2:";  
    byte incomingMsgId = LoRa.read(); // incoming msg ID  
    byte incomingLength = LoRa.read(); // incoming msg length
```

```
    while (LoRa.available()) {  
        incoming += (char)LoRa.read();  
    }
```

```
    if (incomingLength != incoming.length()) { // check length for error  
        //Serial.println("error: message length does not match length");  
        ;  
        return; // skip rest of function  
    }
```

```
    // if the recipient isn't this device or broadcast,  
    if (recipient != Node1 && recipient != MasterNode) {  
        // Serial.println("This message is not for me.");  
        ;  
        return; // skip rest of function  
    }
```

```

if ( sender == 0XBB )
{
    String lat1 = getValue(incoming, ',', 0);
    String long1 = getValue(incoming, ',', 1);
    String ACC1 = getValue(incoming, ',', 2);
    String Gyro1 = getValue(incoming, ',', 3);
    String Fence1 = getValue(incoming, ',', 4);

    latitude1 = lat1.toFloat();
    longitude1 = long1.toFloat();
    acceleration1 = ACC1.toFloat();
    gyroscope1 = Gyro1.toFloat();
    GFence1 = Fence1.toInt();

    incoming = "";
    Serial.print("Data Dari Node 1: ");
    Serial.print(lat1);
    Serial.print(" ");
    Serial.println(long1);
    Serial.print(ACC1);
    Serial.print(" <-> ");
    Serial.println(Gyro1);
    Serial.print("Geo_Fence: ");
    Serial.println(Fence1);

    Firebase.setFloat("latitude1", latitude1);
    Firebase.setFloat("longitude1", longitude1);
    Firebase.setFloat("acceleration1", acceleration1);
    Firebase.setFloat("gyroscope1", gyroscope1);
    //Firebase.setInt("fence1", GFence1);
    delay(200);

}

if ( sender == 0XCC )
{
    String lat2 = getValue(incoming, ',', 0);
    String long2 = getValue(incoming, ',', 1);
    String ACC2 = getValue(incoming, ',', 2);
    String Gyro2 = getValue(incoming, ',', 3);
    String Fence2 = getValue(incoming, ',', 4);

    latitude2 = lat2.toFloat();
    longitude2 = long2.toFloat();
    acceleration2 = ACC2.toFloat();

```

```
gyroscope2 = Gyro2.toFloat();
GFence2 = Fence2.toInt();
```

```
incoming = "";
Serial.print("Data Dari Node 2: ");
Serial.print(lat2);
Serial.print(" ");
Serial.println(long2);
Serial.print(ACC2);
Serial.print(" <-> ");
Serial.println(Gyro2);
Serial.print("Geo_Fence: ");
Serial.println(Fence2);
```

```
Firestore.setFloat("latitude2", latitude2);
Firestore.setFloat("longitude2", longitude2);
Firestore.setFloat("acceleration2", acceleration2);
Firestore.setFloat("gyroscope2", gyroscope2);
//Firestore.setInt("fence2", GFence2);
delay(200);
}
}
```

```
String getValue(String data, char separator, int index)
```

```
{
    int found = 0;
    int strIndex[] = { 0, -1 };
    int maxIndex = data.length() - 1;

    for (int i = 0; i <= maxIndex && found <= index; i++) {
        if (data.charAt(i) == separator || i == maxIndex) {
            found++;
            strIndex[0] = strIndex[1] + 1;
            strIndex[1] = (i == maxIndex) ? i + 1 : i;
        }
    }
    return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```