

BAB I

PENDAHULUAN

A. Latar Belakang

Dalam era modern ini, perkembangan teknologi semakin pesat dan membawa dampak positif bagi berbagai aspek kehidupan manusia. Namun, di sisi lain, perkembangan ini juga menyebabkan berbagai masalah lingkungan, salah satunya adalah masalah sampah. Dalam kehidupan sehari-hari, banyak jumlah sampah yang dihasilkan oleh masyarakat, baik di lingkungan perkotaan maupun pedesaan. Masalah pengelolaan sampah menjadi semakin kompleks, terutama di daerah perkotaan yang padat penduduknya.

Sampah merupakan salah satu dampak negatif terhadap lingkungan, baik dalam konteks perkotaan maupun pedesaan. Walaupun pemerintah dan lembaga lingkungan hidup (BLH) menyediakan tempat-tempat sampah, kesadaran masyarakat dalam membuang sampah pada tempat yang sesuai masih kurang. Fenomena umum yang terjadi adalah ketika tempat sampah sudah penuh, masyarakat cenderung membuang sampah di sekitar area tempat sampah. Praktik ini mengakibatkan pencemaran lingkungan dan menghasilkan aroma yang tidak menyenangkan di lingkungan sekitarnya. Kesadaran masyarakat dalam menjaga kebersihan yang masih rendah terlihat dalam kehidupan masyarakat Indonesia, yang pada akhirnya menjadi salah satu masalah utama yang dapat menyebabkan pencemaran lingkungan akibat pembuangan sampah yang tidak sesuai tempatnya. Semakin bertambahnya jumlah penduduk dan meningkatnya aktivitas di suatu

wilayah berkontribusi pada peningkatan volume sampah. Hal ini mengakibatkan perlunya alokasi biaya yang signifikan serta pengadaan lahan yang semakin luas guna mengatasi masalah sampah.

Banyak siswa yang usai berbelanja saat istirahat memilih untuk bersantai dan berkumpul bersama teman-teman sambil makan dan minum di berbagai tempat. Namun, disayangkan banyak kaleng, botol, dan plastik yang dibuang tidak pada tempatnya atau malah tercampur dalam satu kotak sampah, padahal pihak sekolah hanya menyediakan satu tempat sampah. Kurangnya kesadaran siswa terkait pembuangan sampah telah menyebabkan terjadinya permasalahan sampah yang berserakan secara sembarangan. Praktik pembuangan sampah sembarangan ini membawa dampak negatif bagi lingkungan sekolah, seperti mencemari lingkungan, merusak keindahan visual lingkungan sekolah, meningkatkan polusi udara, dan lain sebagainya.

Dengan adanya tempat sampah Pintar akan menarik perhatian Siswa untuk membuang sampah dengan baik dan dapat memisahkan sampah berdasarkan jenis sampahnya seperti sampah logam (kaleng, besi, baterai), sampah non-logam (plastik) dan Organik agar mudah untuk melakukan pendaaur ulang sampah hingga dapat mengurangi volume sampah.

Ada pun rancangan Alat Pemilah sampah otomatis untuk memisahkan sampah logam (kaleng, besi, baterai), Anorganik (plastik) dan Organik. Pada rancangan ini kami akan mengimplementasikan di sekolah SMKN 2 Pinrang agar siswa maupun guru dan staf dapat menerapkan kebersihan dan mampu mengurangi volume sampah yang ada di lingkungan sekolah.

Perancangan Alat Pemilah sampah otomatis kami menggunakan *Inductive Proximity* sebagai alat sensor untuk jenis sampah Logam, *Nodemcu Esp8266* dan *Arduino Uno* sebagai *mikrokontroler*, Motor *Servo* dan dilengkapi Sensor *Ultrasonik* untuk mengukur volume sampah, memberikan informasi melalui *notifikasi* pada Aplikasi Android.

Rancang bangun ini tentang pengelolaan sampah yang dapat memilah sampah Logam, plastik, organik serta dapat mengirimkan informasi kondisi tempat sampah. Dengan menambahkan fungsi IOT (*Internet Of Things*) yang dapat memberikan informasi tentang keadaan tempat sampah yang sudah penuh kepada pengelolaan sampah menggunakan android.

Berdasarkan dari latar belakang di atas peneliti bermaksud untuk mengangkat penelitian ini dengan **“Rancang Bangun Alat Pemilah Sampah Dengan Menggunakan *Proximity Sensor*”**.

B. Rumusan Masalah

Adapun rumusan masalah yang muncul dari latar belakang adalah sebagai berikut :

1. Bagaimana cara mengakses *Proximity sensor* sebagai alat sensor Pemilah bahan Logam, Anorganik dan Organik?
2. Bagaimana cara mengukur volume sampah menggunakan sensor Ultrasonik?

C. Tujuan Penelitian

Adapun Tujuan penelitian sebagai berikut:

1. Untuk mengetahui cara mengakses *Proximity sensor* sebagai alat sensor Pemilah bahan Logam, Non-Logam dan Organik .
2. Mengetahui cara mengukur volume sampah menggunakan sensor Ultrasonik.

D. Batasan Masalah

Adapun beberapa batasan masalah pada penelitian ini sebagai berikut:

1. Sistem ini tidak dapat mengolah sampah yang tergabung dalam satu bungkus.
2. Jenis sampah yang akan diolah mencakup logam (kaleng, besi, baterai), non-logam (plastik dan kertas), serta organik (sayuran, buah, daun, kayu).
3. Adanya komunikasi antara perangkat keras dan perangkat lunak menggunakan jaringan.
4. Tempat sampah tidak dapat menampung sampah yang berukuran panjang.
5. Sistem tidak mendeteksi dan membedakan jenis sampah yang ada di dalam Ruang sampah.

E. Manfaat Penelitian

Pada penelitian ini memiliki beberapa manfaat sebagai berikut:

1. Manfaat bagi Peneliti
 - a. Peneliti akan memperoleh pemahaman tentang berbagai ilmu yang dapat diterapkan dalam merancang produk dan mampu mengaplikasikannya dalam praktik.

- b. Peneliti dapat menguji dan memahami kinerja tempat sampah otomatis yang menggunakan *Proximity* sensor, serta mendapatkan pengetahuan yang lebih luas untuk menerapkan konsep-konsep yang dipelajari selama perkuliahan.

2. Manfaat bagi Sekolah

- a. Siswa akan lebih sadar akan lingkungan sekitar dan mengambil tindakan dengan membuang sampah sesuai dengan jenisnya.
- b. Meningkatnya minat siswa untuk memilah sampah sesuai jenisnya, mendorong kesadaran lingkungan di kalangan mereka.
- c. Tempat sampah otomatis ini dapat membantu petugas kebersihan dalam proses pemisahan sampah berdasarkan jenis, mempermudah proses daur ulang.

3. Manfaat bagi Institusi

- a. Membangun alat yang mendorong orang lain untuk lebih peduli terhadap lingkungan, karena penggunaan tempat sampah yang tepat akan memberikan poin yang dapat ditukarkan dengan hadiah berguna.
- b. Hasil dari penelitian ini diharapkan dapat menjadi rujukan bagi penelitian serupa atau menjadi landasan untuk pengembangan lebih lanjut.

F. Sistematika Penelitian

Secara garis besar sistematika penulisan yang dapat disajikan sebagai berikut:

BAB I PENDAHULUAN

Bab ini memuat latar belakang permasalahan, rumusan masalah, tujuan dan manfaat penelitian, serta uraian sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini menguraikan teori-teori dasar serta komponen-komponen yang relevan dalam penelitian ini.

BAB III METODE PENELITIAN

Bab ini menjelaskan jenis penelitian, lokasi dan periode penelitian, parameter pengujian, tahapan metodologi penelitian, dan diagram alir penelitian.

BAB IV HASIL PENGUJIAN

Bab ini membahas pengujian perangkat yang terhubung dengan Mikrokontroler serta proses pengumpulan data.

BAB V PENUTUP

Bab ini merangkum kesimpulan dari hasil pengujian yang telah dilakukan, serta memberikan saran-saran untuk perbaikan pada perencanaan di masa yang akan datang.

BAB II

TINJAUAN PUSTAKA

A. Penelitian Terdahulu

Penelitian terdahulu memiliki peran penting dalam memberikan gambaran serta mengkaji kerangka berpikir dalam pembahasan. Selain itu, penelitian ini juga bertujuan untuk mendapatkan data sebagai bahan perbandingan dan acuan. Dalam tinjauan pustaka ini, beberapa hasil penelitian terdahulu yang relevan mengenai analisis sistem kerja alat akan dicantumkan sebagai rujukan.

1. Penelitian oleh (Fatmawati et al., 2020) yang berjudul “Rancang Bangun Tempat Sampah Pintar Menggunakan Sensor Jarak Berbasis Mikrokontroler Arduino” rancang bangun ini Tempat sampah Pintar memiliki dua kompartemen untuk menyimpan sampah, yaitu untuk sampah organik dan sampah anorganik. Terdapat satu pintu masuk untuk memasukkan sampah ke dalam tempat sampah Pintar. Tutup tempat sampah Pintar akan secara otomatis membuka dan menutup dengan mendeteksi kehadiran manusia pada jarak 40 cm. Jika sampah organik terdeteksi, *servo* akan bergerak untuk membuka kompartemen sampah organik. Sebaliknya, jika tidak ada deteksi oleh sensor *Proximity* kapasitif, *servo* akan bergerak untuk membuka kompartemen sampah anorganik. Jika kompartemen tempat sampah telah terisi penuh, maka *buzzer* akan berbunyi dan LED akan menyala sebagai indikasi. Selain itu, sistem juga akan mengirimkan SMS kepada petugas sebagai pemberitahuan bahwa tempat sampah telah mencapai kapasitas penuh.

2. Penelitian oleh (Ibnu Hajar & Jupri, 2021) yang berjudul “Sistem Perancangan Tempat Sampah Logam dan Non Logam dengan menggunakan Aplikasi M.I.T Inventor” Pada penelitian ini telah dirancang sebuah tempat sampah pintar yang dapat mengukur tinggi sampah, memilah sampah logam dan non logam, serta dapat memantau tinggi sampah dan memberikan notifikasi kepada petugas sampah ketika tempat sampah sudah penuh melalui aplikasi android. Pada perancangan ini, sensor ultrasonik digunakan untuk mengukur tinggi sampah. Sedangkan untuk pemilahan sampah logam dan non logam menggunakan sensor *infrared* dan *proximity*. Sistem saling bekerja melalui jaringan internet menggunakan mikrokontroler *NodeMcu* untuk menghubungkan antara rancangan tempat sampah dengan aplikasi android yang dibuat menggunakan MIT *App Inventor*. Berdasarkan pengujian alat dan aplikasi yang telah dilakukan, sensor ultrasonik dapat mengukur tinggi sampah dengan persentase akurasi sebesar 94,07 % dan persentase *error* sebesar 5.93%, sensor *proximity* dapat mendeteksi benda logam pada jarak maksimum 3 mm, serta aplikasi tempat sampah pintar yang dapat bekerja dengan baik sesuai dengan fungsinya.
3. Penelitian oleh (Rachmat Farhan et al., 2019) yang berjudul “Rancang Bangun Tempat Sampah Pintar Pada Gedung Jurusan Teknik Elektro Berbasis Mikrokontroler Arduino Mega 2560” Pada perancangan ini menggunakan sensor *Proximity* induktif untuk mendeteksi sampah logam dan sensor *Proximity* kapasitif untuk mendeteksi sampah non logam basah dan kering. Sensor *ultrasonic* untuk mendeteksi level sampah dari masing-masing penampung sampah dengan jarak deteksi 10 cm. Dilengkapi dengan LCD 16x2

untuk menampilkan jenis sampah yang telah dipilah dan GSM *Module* SIM800L yang akan memberikan notifikasi pemberitahuan kepada petugas kebersihan apabila bak sampah telah penuh melalui *Short Message Service* (SMS). Jika ada sampah yang masuk maka sensor akan mendeteksi jenis sampah kemudian motor *servo* menggerakkan sampah ke arah bak sampah yang sesuai dengan yang telah terdeteksi, kemudian akan tampil jenis sampah pada LCD. Apabila bak sampah terdeteksi sudah penuh, maka akan ada informasi berupa SMS kepada petugas kebersihan untuk membuang sampah. Hasil dari rancang bangun tempat sampah Pintar ini yaitu proses Pemilahan sampah lebih efisien dan memudahkan kinerja petugas kebersihan dalam hal pengangkutan sampah yang sudah penuh tanpa perlu melakukan pengecekan terlebih dahulu, karena jika bak sampah telah penuh maka akan ada informasi bak sampah penuh yang akan dikirim melalui SMS kepada petugas kebersihan.

B. Landasan Teori

1. Tempat sampah

Tempat sampah merupakan wadah yang paling sering digunakan untuk menampung sampah-sampah, yang sering disimpan di dapur, luar ruangan, tempat umum dll.

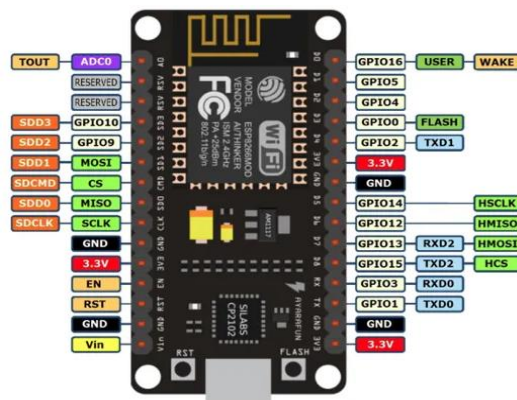
2. NODEMCU ESP8266

ESP8266 adalah modul *transceiver* serial *Wi-Fi*, sebuah komponen *chip* terintegrasi yang dirancang untuk memenuhi kebutuhan dunia modern yang terus terhubung antar perangkat. *Chip* ini memberikan solusi jaringan yang lengkap dan

menyatu, yang dapat digunakan sebagai penyedia aplikasi atau untuk memisahkan semua fungsi *networking* ke proses aplikasi lainnya (Andrianto, 2019).

NODEMCU papan pra-integrasi dengan modul ESP8266 yang dapat digunakan sebagai platform pengembangan *Internet of Things* dan dapat diprogram menggunakan *sketsa open source Arduino IDE Firmware* dan pengembang kit pengembangan membantu Anda membuat prototipe produk IoT Anda dalam beberapa baris skrip Lua. Kit pengembangan berbasis ESP8266, mengintegrasikan GPIO, PWM, IIC, 1-Wire dan ADC dalam satu papan, NODEMCU.com (Safiroh W.P et al., 2022).

NODEMCU adalah sebuah *platform* IoT yang bersifat *open source*. Terdiri dari perangkat keras berupa *System On Chip* ESP8266 dari ESP8266 buatan *Espressif System*, juga *firmware* yang digunakan, yang menggunakan bahasa pemrograman *scripting Lua* (PRATAMA, 2018).



Gambar 2. 1 Nodemcu ESP8266

Tabel 2. 1 Spesifikasi NODEMCU ESP8266

Spesifikasi	Keterangan
Mikrokontroler	Tensilica 32-bit RISC CPU Xtensa LX106
Tegangan operasi	3.3V

Lanjutan Tabel 2.1

Spesifikasi	Keterangan
Tegangan Masukan	7-12V
Pin Digital I/O (DIO)	16
Pin <i>Analog Input</i> (ADC)	1
UARTs	2
SPIs	1
I2Cs	1
<i>Flash Memory</i>	4 MB
SRAM	64 KB
<i>Clock Speed</i>	80 MHz
PCB Antenna	-

3. Arduino Uno

Arduino Uno adalah sebuah papan yang didalamnya terdapat rangkaian elektronik yang bisa diubah atau *open source*, dan di dalam papan tersebut terdapat sebuah *mikrokontroler* berbentuk AVR dari perusahaan *Atmel*. *Mikrokontroler* itu sendiri adalah *chip* atau IC (*integrated circuit*) yang bisa di program menggunakan komputer (Alfian et al., 2021).



Gambar 2. 2 Arduino UNO**Tabel 2. 2** Spesifikasi Arduino Uno

Spesifikasi	Keterangan
Mikrokontroler	<i>Atmega328</i>
Tegangan operasi	5V
Tegangan Masukan	7-11V
<i>I/O Pins</i>	<i>14 Digital (of which 6 provide PWM output), 6 analog</i>
<i>Flash Memory</i>	<i>32 KB (ATmega328) of which 0.5 KB used by bootloader</i>
<i>SRAM</i>	<i>2 KB</i>
<i>EEPROM</i>	<i>1 KB (ATmega328)</i>
Clock Speed	16 MHz

4. Sensor *Proximity*

Sensor *Proximity* adalah jenis sensor atau saklar yang berfungsi berdasarkan jarak antara objek dan sensor itu sendiri. Sensor ini mampu mendeteksi adanya target, terutama jenis logam, tanpa memerlukan kontak fisik. Umumnya, sensor ini terdiri dari komponen elektronik *solid-state* yang dirancang dengan baik untuk melindungi diri dari pengaruh getaran, cairan, bahan kimia, dan kerusakan yang mungkin terjadi. Kelebihan sensor *Proximity* adalah kemampuannya bekerja tanpa perlu menyentuh objek yang dideteksi. Sensor ini dapat diterapkan dalam situasi di mana objek yang terlalu kecil atau lembut untuk mengaktifkan saklar mekanis konvensional.

Terdapat beberapa jenis sensor *Proximity* yang digunakan untuk mendeteksi keberadaan atau ketiadaan suatu objek. Desain dari sensor *Proximity* dapat

didasarkan pada beberapa prinsip operasi, beberapa contohnya meliputi: *variable reluctance*, *eddy current loss*, *saturated core*, dan *hall effect*. Bergantung pada prinsip operasinya, setiap jenis sensor akan memiliki tingkat kinerja yang berbeda dalam mendeteksi berbagai jenis objek. Beberapa jenis umum dari sensor *Proximity* non-kontak meliputi: *Inductive Proximity* Sensor (Sensor Jarak Induktif): Sensor ini menggunakan sirkuit resonansi (osilator) yang mengalami perubahan saat berada di dekat bahan *konduktif*. Perubahan ini menciptakan perubahan dalam kehilangan resistansi sirkuit yang dapat diukur. *Capacitive Proximity* Sensor (Sensor Jarak Kapasitif): Sensor ini mendeteksi perubahan *kapasitansi* yang terjadi saat objek mendekat. Perubahan ini disebabkan oleh perubahan medan *elektrostatik* antara sensor dan objek. *Ultrasonic Proximity* Sensor (Sensor Jarak Ultrasonik): Sensor ini menggunakan gelombang ultrasonik untuk mengukur jarak antara sensor dan objek dengan mengukur waktu pantulan gelombang kembali. *Photoelectric Proximity* Sensor (Sensor Jarak Fotolistrik): Sensor ini menggunakan sinar cahaya untuk mendeteksi objek. Ketika cahaya terhalang oleh objek, sensor akan mendeteksi perubahan cahaya yang diterima.



Gambar 2. 3 *Inductive Proximity Sensor*

Inductive Proximity Sensor adalah jenis sensor jarak yang digunakan khusus untuk mendeteksi keberadaan logam, baik itu logam jenis *Ferrous* maupun non-*ferrous*. Sensor ini efektif dalam mendeteksi apakah ada atau tidak ada objek logam di dekatnya, serta memiliki kemampuan untuk menghitung jumlah objek logam dan juga dapat diterapkan dalam berbagai aplikasi pemosisian.

Tabel 2. 3 *Spesifikasi Inductive Proximity Sensor*

Spesifikasi	Keterangan
<i>Voltase</i>	DC 6-36V
Arus keluaran	300m A
Polaritas	PNP NO (<i>Normally Open</i>)
Dimensi	M12x10x55mm
Jarak Deteksi	4 mm
Status Keluaran	<i>Normally Open</i>
Nomor bagian	LJ12A3-4-Z/BY
Objek deteksi	<i>Conductor</i>
Panjang Kabel	<i>190 cm</i>
Material	<i>Metal and Plastic</i>



Gambar 2. 4 *Sensor Proximity Capacitive*

Sensor *Capacitive Proximity* memiliki kemampuan untuk mendeteksi objek baik logam maupun non-logam. Prinsip kerja dari Sensor *Capacitive Proximity*

adalah dengan cara mengukur perubahan kapasitansi medan listrik pada sebuah kapasitor yang diinduksi oleh adanya objek yang mendekat. Ketika objek mendekat, medan listrik antara sensor dan objek akan mengalami perubahan kapasitansi yang dapat diukur oleh sensor.

Tabel 2. 4 *Spesifikasi Capacitive Proximity Sensor*

Spesifikasi	Keterangan
Tipe	<i>LJC18A3-H-Z/BY</i>
Vin	12-24 vdc
<i>Diameter Head</i>	18 mm
Jarak Sensing	1-10 mm
Dc	3 – wire
Pnp	<i>Normally Open</i>
<i>Led</i>	<i>Red Indikator</i>
<i>Supply Voltage</i>	<i>DC 6 – 36V</i>
<i>Panjang Kabel</i>	120 cm/47"
<i>Response Frequency</i>	100Hz
<i>Detecting Objek</i>	<i>Metal or Nonmetal</i>



Gambar 2. 5 *Sensor Proximity Infrared*

Infrared Proximity Sensor adalah jenis sensor yang mampu mendeteksi keberadaan suatu objek tanpa perlu bersentuhan langsung dengan objek tersebut, melainkan menggunakan teknologi *infrared*. Sensor ini memiliki kemampuan untuk mendeteksi adanya benda yang menghalangi atau tidak, tetapi tidak dapat mengukur jarak ke objek tersebut.

Tabel 2. 5 *Spesifikasi Infrared Proximity Sensor*

Spesifikasi	Keterangan
Type	<i>E18-D80NK</i>
<i>Output Current DC</i>	100mA
<i>Output</i>	<i>DC Three-wire System (NPN)</i>
NPN	<i>Normally Open (O)</i>
<i>Led</i>	<i>Red Indikator</i>
<i>Supply Voltage</i>	<i>DC 5V</i>
<i>sensing range</i>	3-80cm
<i>Response Time</i>	<2ms
<i>Detecting Objek</i>	<i>Transparent or Opaque Body</i>

5. Sensor Ultrasonik

Sensor ultrasonik adalah jenis sensor yang menghasilkan gelombang suara pada frekuensi ultrasonik dan kemudian mendeteksi pantulan gelombang tersebut. Fungsinya adalah untuk mengukur jarak antara sensor dan objek yang ada di depannya. Sensor ini umumnya digunakan dalam aplikasi robotika, terutama dalam pembuatan robot yang dapat menghindari rintangan (*obstacle avoiding robot*).

Sensor ultrasonik dengan kode HC-SR04 memiliki 4 kaki yang masing-masing diberi label *Vcc*, *Trig*, *Echo*, dan *Gnd*. Kaki *Vcc* menandakan bahwa sensor

ini memerlukan tegangan suplai sebesar 5 volt agar dapat beroperasi. Kaki *Trig* digunakan sebagai kaki sinyal pengirim, sementara kaki *Echo* bertanggung jawab dalam menangkap gelombang yang dipantulkan. Kaki *Gnd* merupakan kutub negatif yang digunakan untuk menghubungkan sensor ke tanah atau *ground*. HC-SR04 adalah modul *ultrasonik* yang menyediakan fungsi pengukuran *non*-kontak 2 cm sampai 400 cm. Ketepatan pengukuran jarak mencapai 3mm, (Wafi et al., 2020).



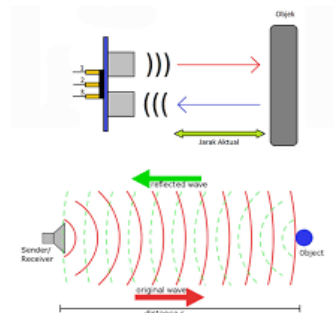
Gambar 2. 6 *Sensor Ultrasonik*

Tabel 2. 6 *Spesifikasi Sensor Ultrasonik*

Spesifikasi	Keterangan
<i>Operating Voltage</i>	3.3V
<i>Operating Current</i>	15mA
<i>Operating Frequency</i>	40KHz
<i>Measuring Range</i>	2 cm – 400cm
<i>Measuring Angel</i>	15 degrees

Pada sensor Ultrasonik, gelombang ultrasonik dihasilkan melalui komponen yang disebut piezoelektrik dengan frekuensi tertentu. Piezoelektrik menghasilkan gelombang ultrasonik (biasanya pada frekuensi 40 kHz) ketika diberikan osilasi. Secara umum, komponen ini mengirimkan gelombang ultrasonik ke suatu area atau target tertentu. Setelah gelombang menyentuh permukaan target, gelombang akan dipantulkan kembali. Gelombang pantulan ini kemudian ditangkap oleh sensor,

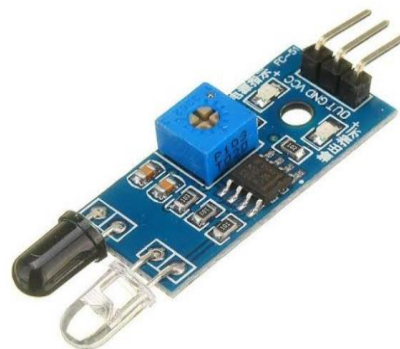
yang menghitung selisih waktu antara pengiriman gelombang dan penerimaan gelombang pantulan.



Gambar 2. 7 Cara Kerja *Sensor Ultrasonik*

6. Sensor *Infrared*

Sebagai alat pendeteksi benda, *Sensor Infrared* adalah komponen elektronik yang berfungsi untuk mendeteksi keberadaan suatu benda saat cahaya inframerah terhalang oleh objek tersebut. Sensor inframerah terdiri dari LED inframerah sebagai sumber cahaya dan *fototransistor* sebagai penerima cahaya inframerah. LED inframerah yang berperan sebagai pemancar cahaya inframerah merupakan singkatan dari "*Light Emitting Diode Infrared*" dan dibuat dari bahan Galium Arsenida (*GaAs*). Ketika diberikan arus listrik, LED ini dapat menghasilkan cahaya inframerah dan radiasi panas. Proses pemancaran cahaya ini sebagai akibat dari energi listrik disebut dengan sifat *elektroluminesensi* (Sutrisno. 1987).



Gambar 2. 8 *Sensor Infrared*

Tabel 2. 7 *Spesifikasi Sensor Infrared*

Spesifikasi	Keterangan
Banyak Pin	3 Pin
Vin	3-5 V
Komunikasi Arus listrik	- 23 mA (3 V) - 43 mA (5 V)
Jarak Deteksi	2 – 30 cm
Sudut Deteksi	35°

7. Motor Servo

Motor *servo* adalah jenis motor DC yang dilengkapi dengan sistem umpan balik tertutup. Motor ini terdiri dari motor DC, rangkaian gigi, kontroler, dan potensiometer. Dalam praktiknya, motor *servo* tidak beroperasi sendirian, melainkan bekerja bersama komponen-komponen lain yang terintegrasi dalam satu kesatuan. Motor *servo* memiliki kemampuan untuk berputar dalam arah searah atau berlawanan jarum jam. Putaran sudut motor *servo* dapat diatur dengan mengontrol pulsa yang masuk ke motor tersebut.

Penting untuk diperhatikan bahwa motor *servo* memerlukan sinyal kontrol yang disebut PWM (*Pulse Width Modulation*) untuk mengatur posisinya. Frekuensi sinyal PWM yang optimal untuk motor servo adalah sekitar 50 Hz, dan kualitas kerjanya sangat tergantung pada frekuensi dan durasi pulsa yang dikirimkan ke *pin* kontrol motor *servo*, (Fatmawati et al., 2020).



Gambar 2. 9 Motor Servo**Tabel 2. 8 Spesifikasi Motor Servo**

Spesifikasi	Keterangan
Merek	Tower pro MG995 180 derajat
<i>Weight</i>	55 g
<i>Dimension</i>	40.7 x 19.7 x 42.9 mm approx.
<i>Stall torque</i>	9.4 kg cm (4.8 V), 11 kg cm (6 V)
<i>Operating speed</i>	0.17 s/60 (4.8 V), 0.14 s/60 (6 V)
<i>Operating voltage</i>	4.8 V a 7.2 V

Lanjutan Tabel 2.8

Spesifikasi	Keterangan
<i>Running Current</i>	500 mA
<i>Stall Current</i>	2.5 A (6V)
<i>Dead band width</i>	5 s
<i>Temperature range</i>	0 C 4.8 V a 7.2 V 900 mA (6V)
<i>double ball bearing design</i>	55 C

8. Kabel Jumper

Jumper pada sebuah komputer sebenarnya adalah penghubung sirkuit elektrik yang digunakan untuk mengatur atau memutuskan hubungan dalam suatu sirkuit. *Jumper* juga berfungsi untuk melakukan pengaturan pada papan induk (*motherboard*) seperti pada *motherboard* komputer. Kabel *jumper* adalah jenis kabel yang umumnya digunakan sebagai penghubung antara *NodeMCU* 8266 dengan papan atau *NodeMCU* 8266 dengan sensor yang akan digunakan. Kabel *jumper* berfungsi untuk mengalirkan listrik atau sinyal. Ini dicapai melalui lapisan

konduktor logam di dalamnya. Terdapat tiga jenis kabel *jumper* yang dapat dibedakan berdasarkan ujungnya:

- a. *Male-Male* (jantan-jantan)
- b. *Male-Female* (jantan-betina)
- c. *Female-Female* (betina-betina)



Gambar 2. 10 Kabel *Jumper*

9. *Power Supply*

Power Supply adalah sebuah komponen yang digunakan untuk memasok atau menyediakan daya listrik ke sebuah atau lebih perangkat.



Gambar 2. 11 *Power Supply*

Tabel 2. 9 *Spesifikasi Power Supply*

Spesifikasi	Keterangan
<i>Output</i>	12V 5A
<i>Input</i>	110 – 220V AC 50/60 Hz
<i>Dimensi</i>	11 x 8 3.7 co

10. Buzzer Sounder

Buzzer sounder adalah perangkat elektronik yang menghasilkan suara berdering atau berbeep ketika diaktifkan. Biasanya digunakan dalam berbagai aplikasi seperti alarm, *timer*, sistem notifikasi, dan permainan elektronik. *Buzzer sounder* umumnya terdiri dari rumah, elemen getar (seperti diafragma atau elemen piezoelektrik), dan rangkaian elektronik yang mengontrol *output* suara.

Suara berdering dihasilkan oleh getaran atau osilasi cepat dari komponen internal ketika arus listrik diterapkan. *Buzzer sounder* dapat bervariasi dalam desain, ukuran, dan tegangan operasionalnya, tergantung pada aplikasi spesifik yang ditujukan.



Gambar 2. 12 Buzzer

Tabel 2. 10 Spesifikasi Buzzer

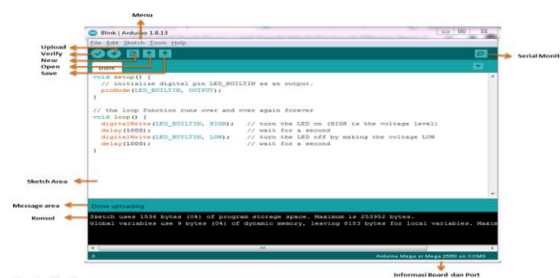
Spesifikasi	Keterangan
<i>model</i>	SFM-27
<i>Rated voltage</i>	12V
<i>voltage range</i>	3~24V
<i>Frequency</i>	3000+/-500Hz
<i>Working temperature</i>	-20~60degree

11. Arduino IDE

IDE merupakan singkatan dari IDE (*Integrated Development Environment*) adalah perangkat lunak yang digunakan untuk mengembangkan program, termasuk

pada Esp8266 *NodeMCU*. Program yang dibuat menggunakan perangkat lunak Arduino IDE disebut sebagai "*sketch*". *Sketch* ditulis dalam editor teks dan disimpan dalam berkas dengan ekstensi tertentu. Pada perangkat lunak Arduino IDE, terdapat kotak pesan berwarna gelap yang berfungsi untuk menampilkan status, seperti pesan kesalahan, kompilasi, dan unggahan program. Di bagian bawah sebelah kanan perangkat lunak Arduino IDE, ditampilkan papan (*board*) yang dikonfigurasi beserta *port* COM yang digunakan.

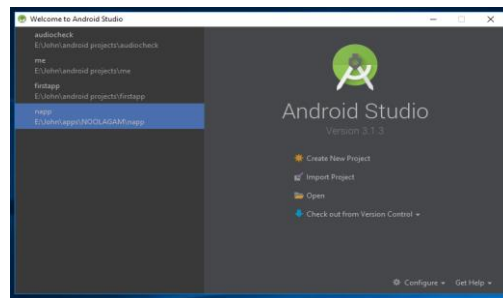
- a. *Verify/Compile*, berfungsi untuk mengecek apakah *sketch* yang dibuat ada kekeliruan dari segi *sintaks* atau tidak. Jika tidak ada kesalahan, maka *sintaks* yang dibuat akan di *compile* ke dalam bahasa mesin.
- b. *Upload*, berfungsi mengirimkan program yang sudah dikompilasi ke Arduino Board (Endra et al., 2019).



Gambar 2. 13 Arduino IDE

12. Android Studio

Android Studio adalah IDE (*Integrated Development Environment*) yang resmi digunakan untuk mengembangkan aplikasi Android, dan memiliki sifat open *source* atau gratis.



Gambar 2. 14 Aplikasi Android Studio

Android studio sendiri dikembangkan berdasarkan *Intellij IDEA* yang mirip dengan *Eclipse* disertai dengan *Android Development Tools (ADT) plugin*. Android studio memiliki fitur :

- a. Proyek berbasis pada *Gradle Build*
- b. *Refactory* dan pembenahan *bug* yang cepat
- c. *Tools* baru yang bernama “*Lint*” diklaim dapat memonitor kecepatan, kegunaan, serta kompatibilitas aplikasi dengan cepat.
- d. Mendukung *Proguard And App-signing* untuk keamanan.
- e. Memiliki GUI aplikasi *android* lebih mudah
- f. Didukung oleh *Google Cloud Platform* untuk setiap aplikasi yang dikembangkan.

13. Java

Java adalah bahasa pemrograman yang dapat dieksekusi di berbagai jenis komputer, termasuk perangkat telepon genggam. Bahasa ini pertama kali diciptakan oleh James Gosling ketika ia bekerja di *Sun Microsystems*, yang saat ini telah menjadi bagian dari *Oracle*. Java pertama kali dirilis pada tahun 1995.

pemrograman berorientasi objek (OOP) yang dapat berjalan pada berbagai platform sistem operasi. Pengembangan Java tidak terbatas pada satu sistem operasi

saja, melainkan dirancang untuk berjalan di berbagai sistem operasi. Java juga bersifat *open source*, yang berarti kode sumbernya dapat diakses dan dimodifikasi oleh siapa saja sesuai dengan lisensi yang berlaku.

1. Sebagian Fitur dari Java

a. *Java Virtual Machine (JVM)*

JVM (*Java Virtual Machine*) adalah suatu mesin maya yang beroperasi dengan meniru perilaku aplikasi pada mesin fisik nyata. JVM memberikan spesifikasi perangkat keras dan *platform* tempat kompilasi kode Java terjadi. Spesifikasi ini memungkinkan aplikasi berbasis Java menjadi independen dari *platform* tertentu, karena proses kompilasi dilakukan oleh JVM. Aplikasi Java dibuat dalam bentuk berkas teks dengan *ekstensi.Java*. Berkas ini kemudian dikompilasi menjadi *bytecode* dalam bentuk berkas dengan *ekstensi.class* atau lebih. *Bytecode* ini berisi serangkaian instruksi yang mirip dengan instruksi kode mesin. Namun, perbedaannya terletak pada fakta bahwa kode mesin perlu dijalankan pada sistem komputer yang sesuai dengan kompilasinya, sedangkan *bytecode* dapat dijalankan pada interpreter Java yang tersedia di berbagai platform sistem komputer dan sistem operasi.

b. *Garbage Collection*

Banyak bahasa pemrograman lain yang mengizinkan seorang *programmer* mengalokasikan memori pada saat dijalankan. Program Java melakukan *garbage collection* yang berarti program tidak perlu menghapus sendiri objek–objek yang tidak digunakan lagi.

c. *Code Security*

Code Security terimplementasi pada Java melalui penggunaan *Java Runtime Environment (JRE)*. Java menggunakan model pengamanan 3 lapis untuk melindungi sistem dari *untrusted Java Code*.

1. Pertama, *classloader* menangani pembuatan kelas Java ke *runtime interpreter*. Proses ini menyediakan pengamanan dengan memisahkan kelas-kelas yang berasal dari *local disk* dengan kelas-kelas yang diambil dari jaringan. Hal ini membatasi aplikasi *Trojan* karena kelas-kelas yang berasal dari *local disk* yang dimuat terlebih dahulu.
2. Kedua, *bytecode verifier* membaca *bytecode* sebelum dijalankan dan menjamin *bytecode* memenuhi aturan-aturan dasar bahasa Java.
3. Ketiga, manajemen keamanan menangani keamanan tingkat aplikasi dengan mengendalikan apakah program berhak mengakses sumber daya seperti sistem *file*, *port* jaringan, proses eksternal dan sistem *windowing*.

d. Fase-fase pemrograman Java

Editor adalah perangkat lunak yang digunakan untuk membuat atau mengedit kode program Java.

1. *Hello.java* adalah nama berkas Java yang sudah berisi kode-kode Java dan disimpan dengan *ekstensi.java*.
2. *Java compiler* adalah perangkat yang mengubah berkas Java dengan *ekstensi .java* menjadi berkas dengan *ekstensi.class*.
3. *Hello.class* adalah hasil keluaran dari proses kompilasi berkas Java, yang awalnya memiliki *ekstensi.java*.

4. Java interpreter adalah perangkat yang menerjemahkan baris per baris kode program ke dalam bahasa mesin dan menampilkan hasilnya.



Gambar 2. 15 Bahasa Pemrograman Java

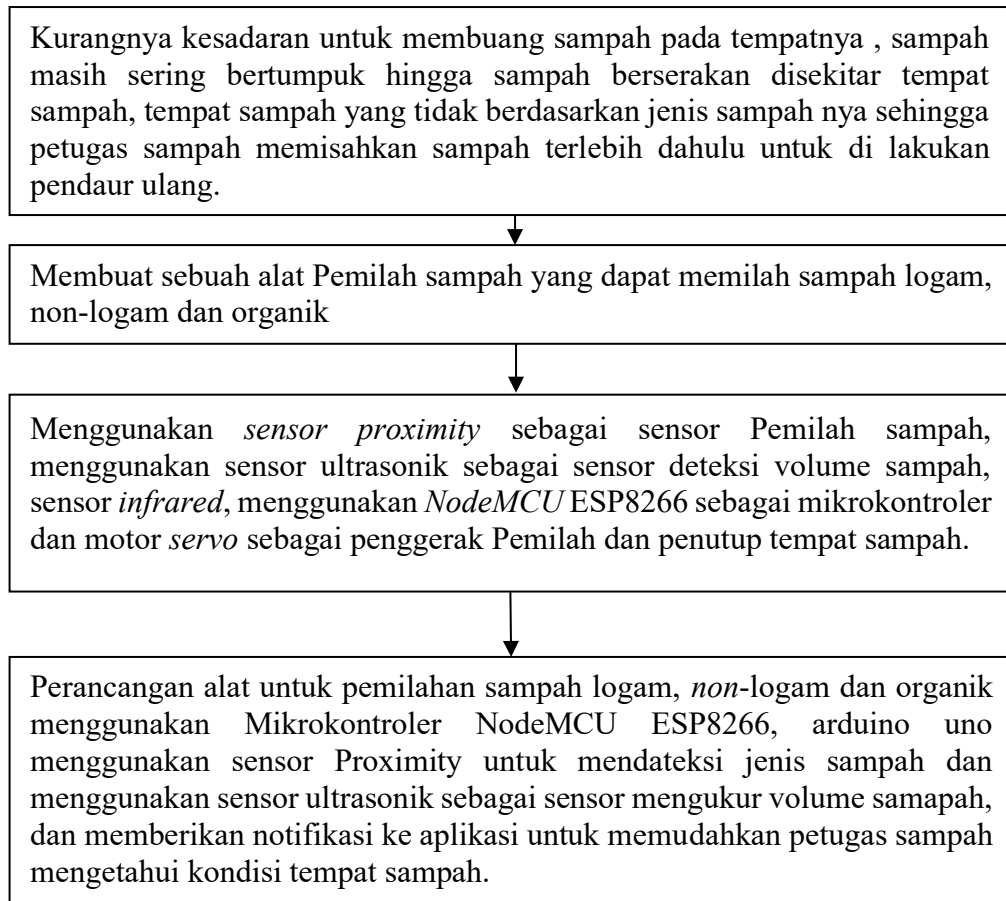
14. Firebase

Realtime Database dikirimkan dengan SDK seluler dan web sehingga Anda dapat mem-*build* aplikasi tanpa memerlukan server. Anda juga dapat mengeksekusi kode *backend* yang merespons peristiwa yang dipicu oleh *database* Anda menggunakan *Cloud Functions for Firebase*. *Realtime Database* terintegrasi dengan *Firebase Authentication* untuk menyediakan proses *autentikasi* yang mudah dan intuitif bagi developer. Anda dapat menggunakan model keamanan deklaratif kami untuk mengizinkan akses berdasarkan identitas pengguna atau pencocokan pola pada data Anda.



Gambar 2. 16 *Firebase*

C. KERANGKA PIKIR



Gambar 2. 17 Kerangka Pikir

BAB III

METODE PENELITIAN

A. Jenis Penelitian

Penelitian ini merupakan Eksperimen atau percobaan yang dilakukan secara terencana, sistematis, dan terkontrol dengan ketat, baik dalam bentuk desain fungsional maupun desain faktorial.

B. Waktu Penelitian

Waktu yang dipergunakan untuk pelaksanaan penelitian ini selama ± 3 (tiga) bulan, dan lokasi penelitian Lab.TKJ SMKN 2 PINRANG.

C. Metode Pengumpulan Data

Untuk mendapatkan data yang diperlukan sebagai dasar dalam penelitian, penulis melakukan pengumpulan data dari lapangan dengan menggunakan tiga metode berikut:

1. Observasi: Observasi adalah proses pengamatan langsung terhadap peristiwa di lapangan, yang memungkinkan penulis untuk memperkuat data yang telah ada.
2. Dokumentasi: Dokumentasi merupakan teknik pengumpulan data dengan menghimpun berbagai dokumen yang tersedia.
3. Studi Literatur: Studi literatur dilakukan untuk memperoleh pemahaman mengenai otomatisasi menggunakan Arduino Uno, NODEMCU ESP8266, dan *Proximity* Sensor sebelumnya.

Pengetahuan yang digunakan untuk merumuskan kaidah atau aturan diperoleh melalui sumber-sumber seperti buku, jurnal, dan artikel.

D. Alat dan Bahan Penelitian

Adapun alat yang digunakan terdapat pada Tabel :

Tabel 3. 1 Bahan

No.	Nama Bahan	Volume
1	<i>Nodemcu Esp8266</i>	1 Unit
2	Arduino Uno	1 Unit
3	<i>Induktif Proximity Sensor</i>	1 Unit
4	<i>Capacitive Proximity Sensor</i>	1 Unit
5	<i>Infrared Proximity Sensor</i>	1 Unit
6	<i>Sensor Ultrasonik</i>	3 Unit
7	<i>Sensor Infrared</i>	1 Unit
8	Motor Servo	3 Unit
9	Kabel Power	1 Unit
10	Kabel Jumper	3 Pasang
11	<i>Power Supply</i>	1 Unit
12	<i>Buzzer Alarm</i>	1 Unit

Tabel 3. 2 Alat

No.	Nama Alat	Volume
1	Solder	1 Unit
2	Laptop	1 Unit (Acer, OS Windows 10 Pro, Intel(R) Celeron(R) CPU 1019Y @ 1.00GHz, RAM 4,00 GB, SSD 120GB)

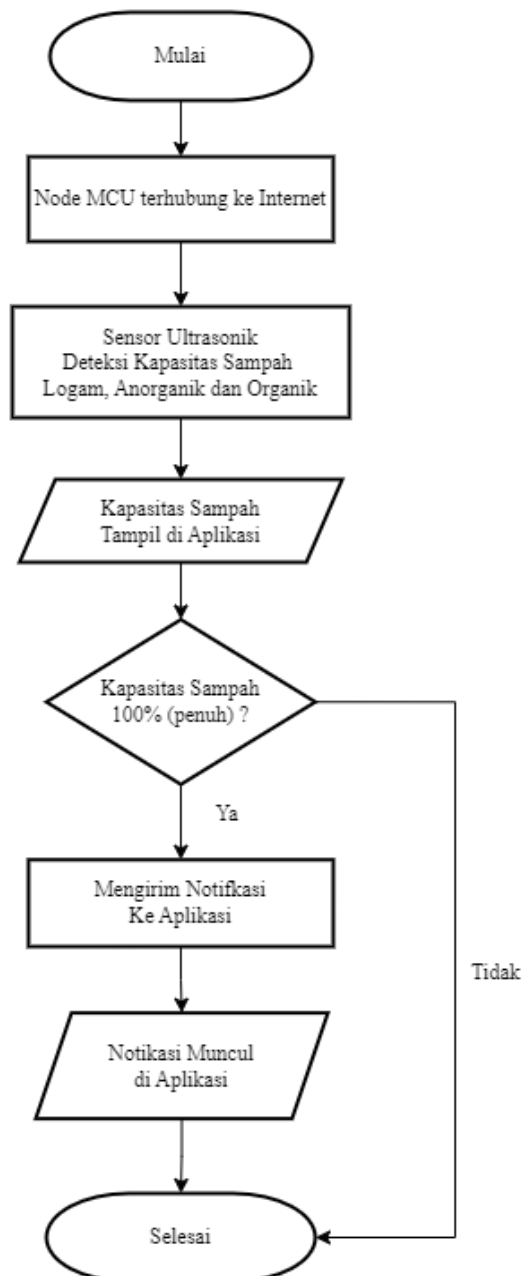
Tabel 3. 3 Perangkat Lunak

No.	Perangkat Lunak
1	Arduino IDE
2	Android Studio

BAB IV

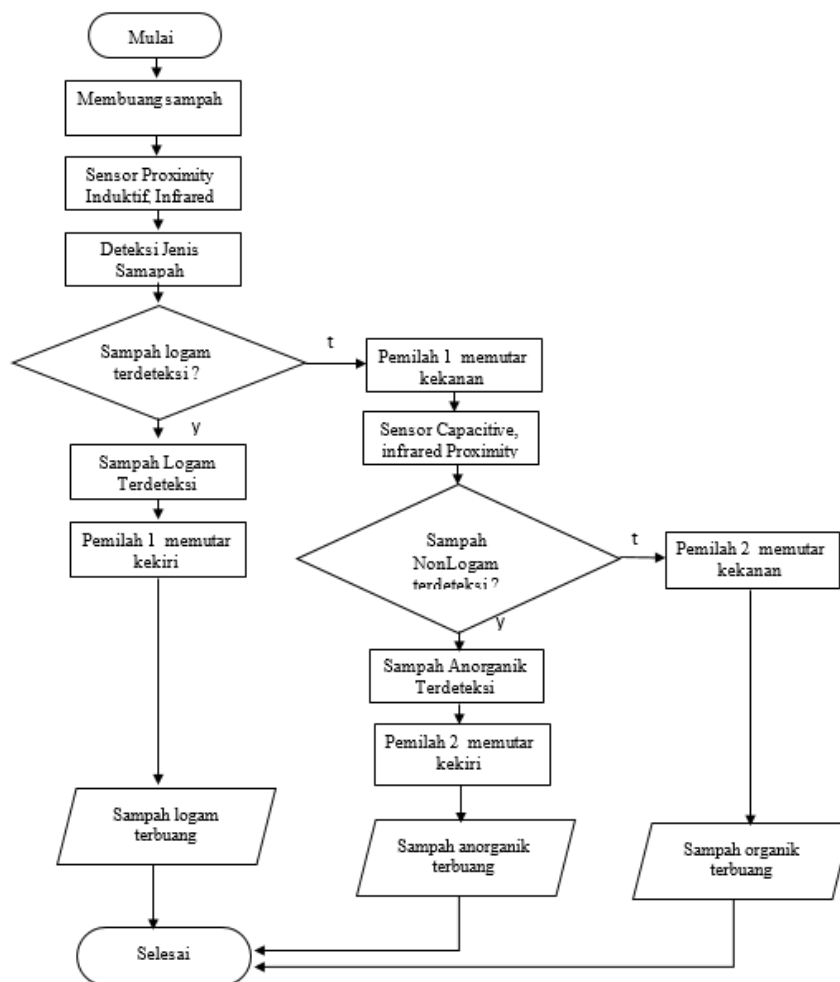
HASIL DAN PEMBAHASAN PENELITIAN

A. Sistem Yang Diusulkan



Gambar 4. 1 Sistem *Monitoring* Yang Diusulkan

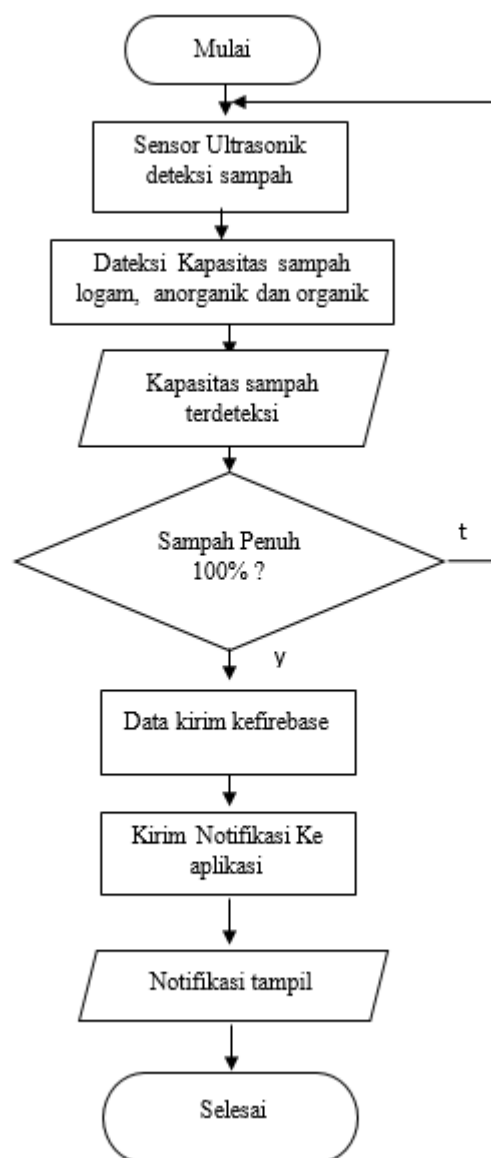
Pada sistem yang diusulkan untuk petugas untuk mengambil sampah yang kapasitasnya sudah penuh, sistem ini akan memberikan informasi ke petugas melalui aplikasi *monitoring* informasi ini akan muncul apabila sensor ultrasonik mendeteksi kapasitas sampah dan NODEMCU akan mengirim data ke *firebase* dari *firebase* akan mengirim data ke aplikasi.



Gambar 4. 2 Flowchart sistem pemilah sampah otomatis

- Pengguna pertama-tama membuang sampah.
- Sensor induktif dan *infrared* mendeteksi jenis sampah yang dibuang.
- Jika sensor mendeteksi sampah logam, motor *servo* berputar ke kiri, dan sampah langsung dibuang ke tong sampah logam.

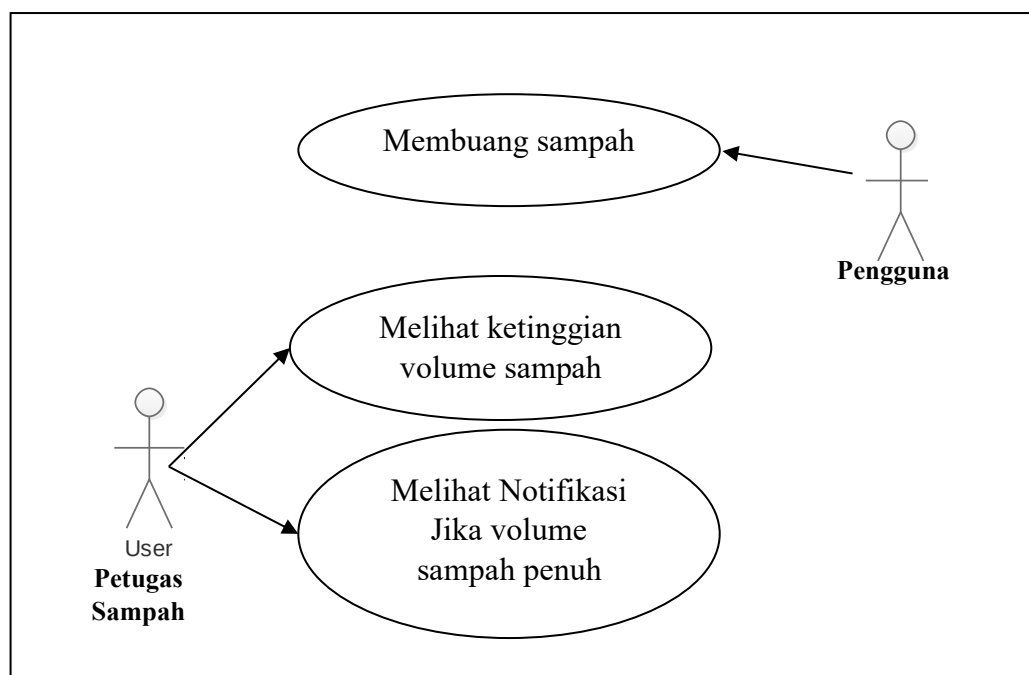
- d. Jika sensor mendeteksi sampah non-logam, motor *servo* berputar 180° ke Pemilah 2 (dua).
- e. Sensor kapasitif dan sensor *infrared* mendeteksi sampah anorganik, lalu motor *servo* berputar ke kiri, dan sampah anorganik dibuang.
- f. Jika sampah anorganik tidak terdeteksi oleh sensor, motor *servo* tetap berputar ke kiri, dan sampah organik dibuang



Gambar 4. 3 Flowchart Deteksi Volume Sampah dan Notifikasi

Pada penjelasan *flowchart* deteksi Kapasitas sampah ini yang pertama sensor ultrasonik deteksi kapasitas sampah logam, anorganik dan organik, jika sampah penuh 100% maka *nodemcu* mengirim data ke *firebase*, kirim notifikasi ke aplikasi dan notifikasi akan tampil, jika kapasitas tidak 100% maka kembali mendeteksi kapasitas sampah.

B. Use Case



Gambar 4. 4 Use Case Aplikasi Android

Pada **Gambar 4.3** Use Case diatas adalah Use Case aplikasi dimana petugas melihat kapasitas sampah dan melihat notifikasi jika volume sampah penuh.

C. Perancangan Desain

a. Tampilan *Monitoring Trash Bin*

Tampilan ini merupakan keadaan kapasitas sampah Logam 0%, Non-Logam 0%, organik 0%.



Gambar 4. 5 Tampilan Aplikasi *Monitoring Trash Bin*

b. Tampilan Notifikasi

Tampilan ini merupakan tampilan notifikasi jika sensor mendeteksi kapasitas sampah penuh dan akan mengirim kan ke aplikasi.



Gambar 4. 6 Tampilan Notifikasi

D. Perancangan Perangkat Keras (*Hardware*)

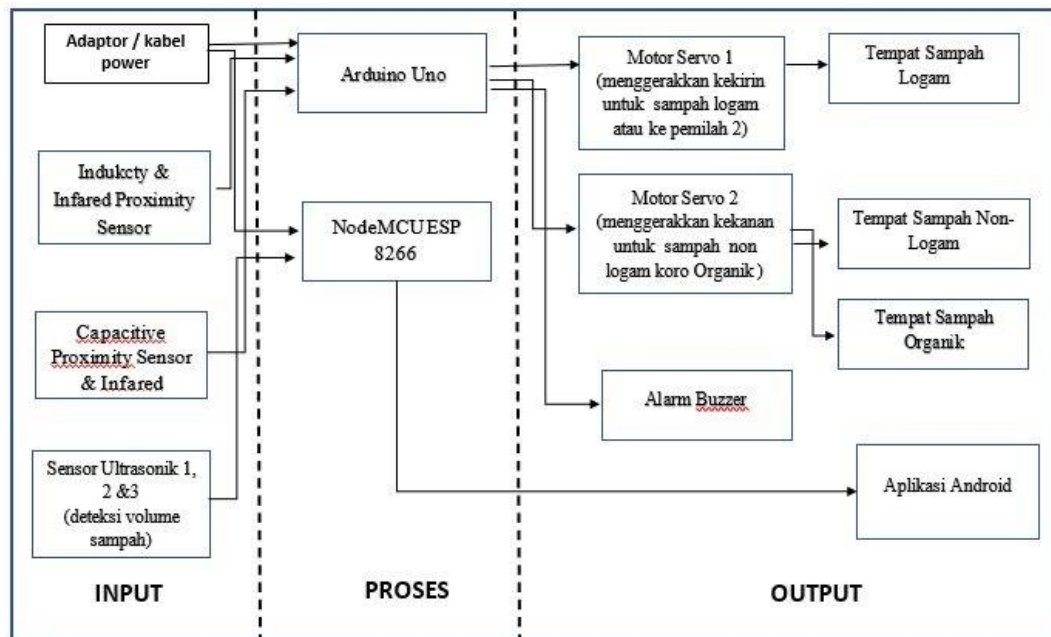
Pembuatan perangkat keras adalah langkah awal sebelum terbentuknya suatu sistem, termasuk rangkaian elektronik. Dalam rancangan ini, pemilihan alat dan komponen yang sesuai sangat penting. Prinsip dasar dalam rancangan alat Pemilah Sampah Otomatis ini melibatkan perangkat keras yang meliputi *Arduino Uno*, *NODEMCU ESP8266*, *Sensor Proximity Capacitive*, *Sensor Proximity Induktif*, *Sensor Proximity Infrared*, *Sensor Ultrasonik*, *Sensor Infrared*, *Motor Servo*, dan *Power Supply*.

E. Implementasi Perangkat Keras (*Hardware*)

1. Rangkaian Perancangan Alat Pemilah Sampah

Blok Diagram dibawa terdapat bagian perangkat keras (*Hardware*) yaitu sebagai berikut :

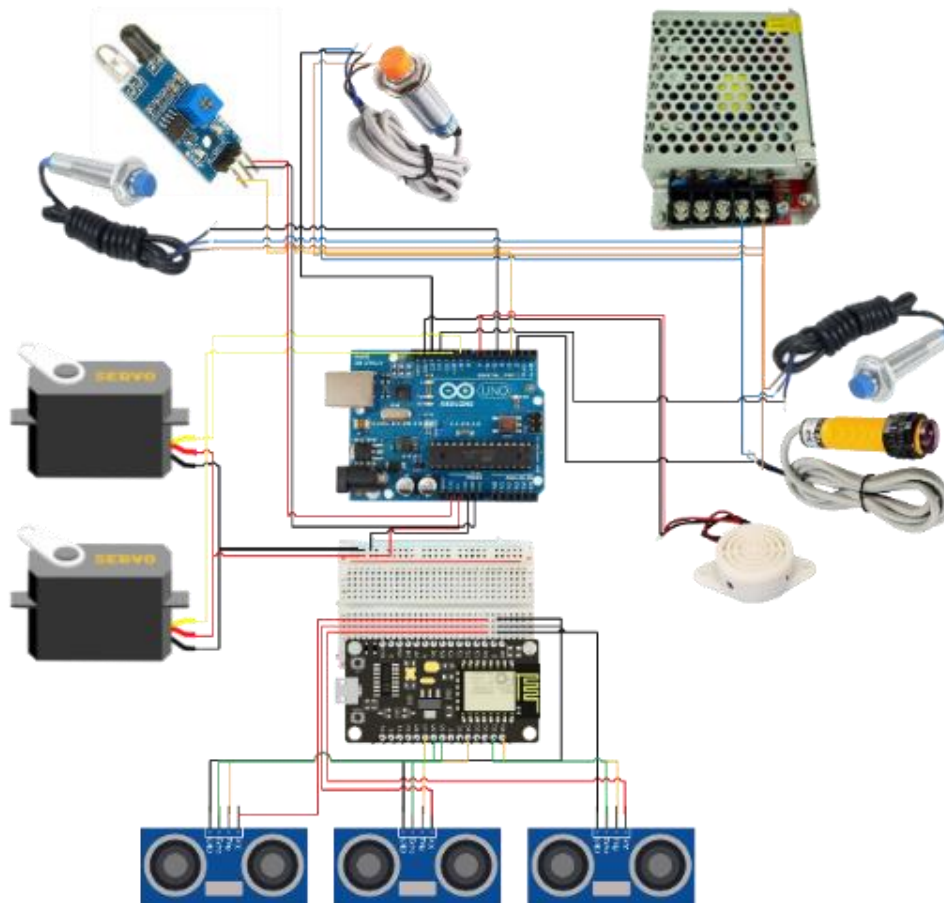
1. *Arduino Uno*
2. *Sensor Proximity Induktif*
3. *Sensor Proximity Kapasitif*
4. *Sensor Proximity Infrared*
5. *Motor servo*
6. *NODEMCU Esp8266*
7. *Sensor Ultrasonik*
8. *Sensor infrared*
9. *Buzzer*



Gambar 4. 7 Block Diagram Rangkaian Alat

- a. *Input* : terdiri dari beberapa perangkat yang terhubung ke *mikrokontroler* (*adaptor, inductive proximity sensor, Capacitive proximity sensor, Infrared proximity sensor, sensor ultrasonik, sensor infrared*) perangkat ini termasuk alat *input*.
- b. *Proses* : kami menggunakan NODEMCU ESP8266 *Mikrokontroler* sebagai alat proses dan mengirim informasi kapasitas sampah ke android dan Arduino Uno sebagai *Mikrokontroler* Pemilah sampah.
- c. *Output* : kami menggunakan perangkat motor *servo* sebagai media pembuka tutup tempat sampah, tempat sampah, dan aplikasi Android, alarm *Buzzer*.

Adapun Alur Pengkabelan atau rangkaian perancangan alat Pemilah sampah menggunakan *Sensor Proximity*.



Gambar 4. 8 Rangkaian Perangkat Keras

Adapun Uraian masing-masing perangkat keras yaitu :

a. Arduino Uno

Arduino Uno diAnalogikan sebagai *Board Arduino* yang terhubung ke perangkat lainnya. Adapun Pin yang digunakan untuk *Arduino Uno* yaitu:

Tabel 4. 1 Penggunaan *Pin Arduino Uno*

No.	Pin / Soket Arduino Uno	Keterangan
1	13	Dihubungkan ke <i>Output</i> Sensor Kapasitif
2	12	Dihubungkan ke <i>Output</i> Sensor Induktif
3	10	Dihubungkan ke data Motor servo
4	9	Dihubungkan ke data Motor servo

Lanjutan Tabel 4.1

No.	Pin / Soket Arduino Uno	Keterangan
5	3	Dihubungkan ke <i>Output Sensor Proximity Infrared</i>
6	2	Dihubungkan ke <i>Output Sensor Proximity Infrared</i>
7	G	Dihubungkan ke GND Motor Servo
8	5V	Dihubungkan ke VCC Motor Servo

b. Sensor Proximity Induktif

Sensor Proximity Induktif merupakan sensor jarak yang sering digunakan untuk deteksi jenis benda dan *Sensor Proximity Induktif* ini digunakan untuk deteksi logam. Adapun Pin yang digunakan untuk *Sensor Proximity Induktif* yaitu:

Tabel 4. 2 Penggunaan *Pin Sensor Proximity Induktif*

No.	Pin Sensor <i>Proximity</i> induktif	Keterangan
1	Output	Dihubungkan ke Pin 12 Arduino Uno
2	V+	Dihubungkan Ke V+ Power Supply
3	V-	Dihubungkan Ke V- Power Supply

c. Sensor Proximity Kapasitif

Sensor Proximity Kapasitif digunakan untuk mendeteksi jarak antara sensor dan objek. Sensor ini umumnya digunakan untuk mengenali jenis benda, dan khususnya digunakan untuk mendeteksi benda berbasis logam dan non-logam seperti plastik, kaca, dan lain sebagainya. Sensor ini memiliki beberapa *pin* yang digunakan untuk pengaturan dan operasinya, seperti:

Tabel 4. 3 Penggunaan *Pin Sensor Proximity Kapasitif*

No.	Pin Sensor <i>Proximity Kapasitif</i>	Keterangan
1	<i>Output</i>	Dihubungkan ke Pin 13 Arduino Uno
2	V+	Dihubungkan Ke V+ <i>Power Supply</i>
3	V-	Dihubungkan Ke V- <i>Power Supply</i>

d. *Sensor Proximity Infrared*

Sensor *Proximity Infrared* merupakan sensor jarak yang sering digunakan untuk deteksi jenis benda yang memantulkan cahayanya kembali. Adapun *Pin* yang digunakan untuk *Sensor Proximity Infrared* yaitu:

Tabel 4. 4 Penggunaan *Pin Sensor proximity infrared*

No.	Pin Sensor <i>Proximity Infrared</i>	Keterangan
1	<i>Output</i>	Dihubungkan ke <i>Pin 2</i> dan <i>Pin 3</i> Arduino Uno
2	V+	Dihubungkan Ke V+ <i>Power Supply</i>
3	V-	Dihubungkan Ke V- <i>Power Supply</i>

e. *Motor Servo*

Motor *servo* merupakan perangkat listrik yang digunakan untuk mendorong dan memutar objek Pada penelitian ini menggunakan 2 (Dua) Motor *Servo* sebagai penggerak Pemilah sampah. Adapun *Pin* yang digunakan pada motor *servo* yaitu:

Tabel 4. 5 Penggunaan *Pin Motor Servo*

No.	Pin Motor <i>Servo</i>	Keterangan
1	PWM	Dihubungkan ke <i>Pin 9</i> dan <i>Pin 10</i> Arduino Uno
2	+5V	Dihubungkan Ke V5 Arduino Uno
3	GND	Dihubungkan Ke GND Arduino Uno

f. *Nodemcu* ESP8266

NODEMCU Esp8266 dianalogikan sebagai *board arduino* yang terkoneksi dengan Esp8266. *Nodemcu* juga sudah terintegrasi dengan berbagai fitur selayaknya *mikrokontroler* dan kapasitas aksesnya terdapat *wifi* dan *chip* komunikasi yang berupa *USB to serial* sehingga pemrograman hanya membutuhkan kabel data *USB*. Adapun Pin yang digunakan untuk *NODEMCU Esp8266* yaitu :

Tabel 4. 6 Penggunaan *Pin/Soket Nodemcu ESP866*

No.	Pin / Soket Nodemcu ESP8266	Keterangan
1	VU	Dihubungkan ke VCC Sensor Ultrasonik 1, 2, & 3
2	D0	Dihubungkan Ke TRIG Sensor Ultrasonik 3
3	D1	Dihubungkan ke ECHO Sensor Ultrasonik 3
4	D4	Dihubungkan Ke TRIG Sensor Ultrasonik 1
5	D5	Dihubungkan ke ECHO Sensor Ultrasonik 1
6	D6	Dihubungkan Ke ECHO Sensor Ultrasonik 2
7	D7	Dihubungkan ke TRIG Sensor Ultrasonik 2
8	GND	Dihubungkan Ke GND Sensor Ultrasonik 1,2 & 3

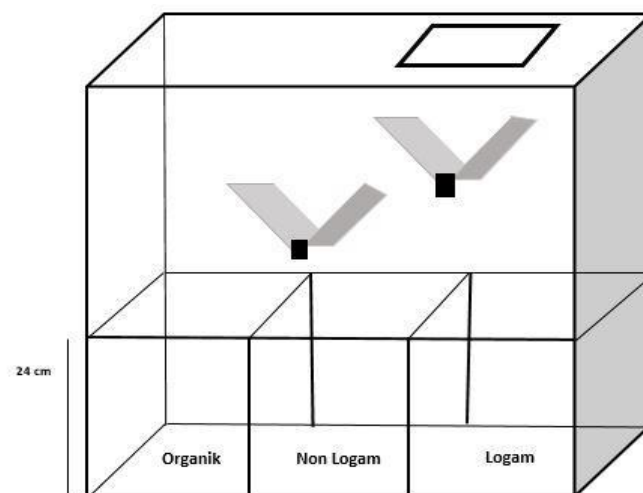
g. Sensor Ultrasonik

Sensor Ultrasonik berfungsi untuk deteksi jarak, Sensor ini bekerja dengan mengirimkan gelombang ultrasonik. Gelombang ini dipantulkan kembali oleh objek dan sensor mendeteksinya. Adapun min yang digunakan untuk Sensor Ultrasonik ini yaitu :

Tabel 4. 7 Penggunaan *Pin Sensor Ultrasonik*

No.	Pin Sensor Ultrasonik	Keterangan
1	VCC	Dihubungkan ke Pin VU/V5 NODEMCU Esp8266
2	ECHO	Dihubungkan ke D1, D5 dan D6 NODEMCU Esp8266
3	TRIG	Dihubungkan ke D0, D4 & D7 NODEMCU Esp8266
4	GND	Dihubungkan Ke GND NODEMCU Esp8266

2. Desain tempat sampah

**Gambar 4. 9** Desain Tempat Sampah Logam, Anorganik, Organik**Gambar 4. 10** Rangka *Box* Pemilah Sampah



Gambar 4. 11 Penempatan Sensor *Proximity* di Dalam Box

F. Rancangan Perangkat Lunak (*Software*)

Setelah melakukan rancangan perangkat keras selanjutnya melakukan perancangan perangkat lunak. Pada perangkat lunak ini merupakan bagian pembuatan program yang akan di *upload* ke dalam *mikrokontroler* yang berupa perintah untuk mengelola data untuk diaplikasikan di perangkat keras. Adapun bahasa program yang digunakan adalah *Sketch* program dalam bahasa C dan untuk pembuatan aplikasi *Monitoring Trash Bin* menggunakan pemrograman *java*. Dan untuk mengirim informasi dari perangkat keras ke aplikasi android menggunakan *Firebase* sebagai *Realtime Database*.

```
Tempat_pemilahsampah
#include <Servo.h>
#include <Wire.h>
```

Gambar 4. 12 Kode untuk menambahkan pustaka

Kode `#include <Servo.h>` dan `#include <Wire.h>` digunakan untuk menambahkan pustaka yang membantu program bekerja dengan perangkat tertentu. *Servo.h* dipakai untuk mengontrol motor *servo*, misalnya memutarnya ke sudut tertentu seperti 0° hingga 180°. *Wire.h* digunakan untuk komunikasi I²C, yaitu cara *mikrokontroler* berbicara dengan perangkat lain, seperti sensor atau layar. Pustaka ini mempermudah pengendalian perangkat tanpa harus menulis kode yang rumit.

```

int Alarm = 8; //variabel Alarm

Servo Servo_Induktif; //Servo Induktif
Servo Servo_Kapasitif; //Servo Kapasitif
#define Sensor_Induktif 12 //pin 12
#define Sensor_Kapasitif 13 //pin 13
#define Sensor_Optical_1 2 //Pemilah infrare pin 2
#define Sensor_Optical_2 3 //Pemilah infrare pin 3
#define Sensor_Alarm 5

```

Gambar 4. 13 Mendeklarasikan Beberapa Variabel

Pada gambar di atas adalah kode untuk mendeklarasikan beberapa variabel, objek, dan *pin* untuk menghubungkan komponen elektronik dengan mikrokontroler. Variabel Alarm disimpan di *pin* 8 (delapan), digunakan untuk mengontrol fitur alarm. Objek *Servo_Induktif* dan *Servo_Kapasitif* dibuat untuk mengontrol motor *servo* yang akan menggerakkan mekanisme tertentu. Pin 12 (dua belas) dan 13 (tiga belas) didefinisikan sebagai *input* dari Sensor Induktif dan Sensor Kapasitif, yang mungkin digunakan untuk mendeteksi jenis material (seperti logam atau plastik). Pin 2 dan 3 digunakan untuk Sensor *Optical* 1 (satu) dan Sensor *Optical* 2 (dua), yang berfungsi sebagai sensor inframerah untuk memilah objek. Terakhir, *pin* 5 (lima) didefinisikan sebagai Sensor Alarm, yang mungkin digunakan untuk memberi peringatan jika terjadi kondisi tertentu.

```

bool Value_Induktif;
bool Value_Kapasitif;
bool Value_Optical_1;
bool Value_Optical_2;

```

Gambar 4. 14 Mendefinisikan Empat Variabel Dengan Tipe Bool

Kode di atas untuk mendefinisikan empat variabel dengan tipe *bool* (*boolean*), yaitu *Value_Induktif*, *Value_Kapasitif*, *Value_Optical_1*, dan *Value_Optical_2*. Variabel ini digunakan untuk menyimpan hasil pembacaan dari sensor-sensor yang hanya memiliki dua kemungkinan nilai: *true* (benar) jika sensor mendeteksi sesuatu, atau *false* (salah) jika tidak ada yang terdeteksi. *Value_Induktif* menyimpan status dari sensor induktif untuk mendeteksi logam, sedangkan *Value_Kapasitif* menyimpan status sensor kapasitif untuk mendeteksi material non-logam. *Value_Optical_1* dan *Value_Optical_2* digunakan untuk menyimpan hasil pembacaan dari sensor optik, seperti mendeteksi keberadaan objek. Semua variabel ini membantu program mengambil keputusan berdasarkan kondisi sensor.

```
void setup() {
    Serial.begin(115200);
    pinMode(Sensor_Induktif, INPUT);
    pinMode(Sensor_Kapasitif, INPUT);
    pinMode(Sensor_Optical_1, INPUT);
    pinMode(Sensor_Optical_2, INPUT);
    pinMode(Sensor_Alarm, INPUT);
    pinMode(Alarm, OUTPUT);

    Servo_Induktif.attach(9); //pin 9
    Servo_Kapasitif.attach(11); //pin 11
}
```

Gambar 4. 15 Fungsi *Setup()*

Kode ini adalah bagian dari fungsi *setup()*, yang dijalankan sekali saat program dimulai. Di dalamnya, *Serial.begin(115200)* digunakan untuk memulai komunikasi serial dengan kecepatan 115200 *baud*. Selanjutnya, *pinMode()* mengatur pin-pin tertentu sebagai *input* atau *output*. Pin sensor (seperti *Sensor_Induktif*, *Sensor_Kapasitif*, dan lainnya) diatur sebagai *INPUT* untuk membaca data dari sensor, sementara pin Alarm diatur sebagai *OUTPUT* untuk

mengendalikan alarm. Terakhir, *Servo_Induktif.attach(9)* dan *Servo_Kapasitif.attach(11)* menghubungkan motor *servo* ke pin 9 (sembilan) dan 11 (sebelas), masing-masing, untuk mengontrol pergerakan mereka.

```
void loop() {
    bool Value_Alarm;

    Value_Induktif = digitalRead(Sensor_Induktif);
    Value_Kapasitif = digitalRead(Sensor_Kapasitif);
    Value_Optical_1 = digitalRead(Sensor_Optical_1);
    Value_Optical_2 = digitalRead(Sensor_Optical_2);
    Value_Alarm = digitalRead(Sensor_Alarm);
```

Gambar 4. 16 Fungsi *Loop()*,

Kode ini adalah bagian dari fungsi *loop()*, yang berjalan terus-menerus selama program aktif. Di dalamnya, status sensor-sensor seperti Induktif, Kapasitif, *Optical 1* (satu), *Optical 2* (dua), dan Alarm dibaca menggunakan *digitalRead()*, yang mengembalikan nilai *HIGH* (terdeteksi) atau *LOW* (tidak terdeteksi). Nilai yang dibaca dari sensor tersebut disimpan dalam variabel yang sesuai (*Value_Induktif*, *Value_Kapasitif*, dll.) untuk digunakan dalam logika program selanjutnya, misalnya untuk mengendalikan motor *servo* atau alarm.

```
| delay(5000);
if (Value_Induktif == 1 && Value_Optical_1 == 0) {
    delay(1000);
    //cek console untuk deteksi jenis sampah
    Serial.print("LOGAM ==> ");
    Serial.print(Value_Induktif);
    Serial.print(" OPTIK ==> ");
    Serial.println(Value_Optical_1);

    //putar servo ke tong logam
    Servo_Induktif.write(0); //buang
    delay(1000);
    Servo_Induktif.write(90); //kembalikan posisi
```

Gambar 4. 17 Memeriksa Kondisi Sensor Pemilah Sampah Logam

Kode ini berfungsi untuk mendeteksi objek sampah menunggu selama 5 detik dengan *delay*(5000) sebelum memeriksa kondisi sensor. Jika sensor Induktif mendeteksi logam (*Value_Induktif* == 1) dan sensor *Optical* 1 tidak terdeteksi objek (*Value_Optical_1* == 0), maka program akan mencetak informasi ke konsol serial untuk menunjukkan deteksi jenis sampah. Setelah itu, motor *servo* yang terhubung ke *Servo_Induktif* akan bergerak. *Servo* akan diputar ke posisi 0 derajat untuk membuang sampah logam, kemudian setelah 1 detik, *servo* dikembalikan ke posisi 90 derajat untuk posisi awal.

```

} if (Value_Induktif == 0 && Value_Optical_1 == 0) {
    delay(1000);
    //cek console untuk deteksi jenis sampah
    Serial.print("NONLOGAM ==> ");
    Serial.print(Value_Induktif);
    Serial.print(" OPTIK ==> ");
    Serial.println(Value_Optical_1);

    //putar servo ke pemilah 2
    Servo_Induktif.write(180); //buang ke pemilah 2
    delay(1000);
    Servo_Induktif.write(90); //kembalikan posisi

```

Gambar 4. 18 Memeriksa Kondisi Sensor Pemilah Sampah Non-Logam

Kode ini memeriksa kondisi sensor setelah 1 detik dengan *delay*(1000). Jika sensor Induktif tidak mendeteksi logam (*Value_Induktif* == 0) dan sensor *Optical* 1 (satu) juga tidak mendeteksi objek (*Value_Optical_1* == 0), maka program akan mencetak informasi ke konsol serial untuk menunjukkan bahwa sampah yang terdeteksi adalah non-logam. Setelah itu, motor *servo* yang terhubung ke *Servo_Induktif* akan diputar ke posisi 180° untuk membuang sampah ke Pemilah 2 (dua), lalu setelah 1(satu) detik, *Servo* dikembalikan ke posisi 90° untuk posisi awal.

```

if (Value_Optical_2 == 0 && Value_Kapasitif == 1) {
    delay(1000);
    //cek console untuk deteksi jenis sampah
    Serial.print("ORGANIK ==> ");
    Serial.print(Value_Kapasitif);
    Serial.print(" OPTIK ==> ");
    Serial.println(Value_Optical_2);

    /*putar servo ke tong organik*/
    Servo_Kapasitif.write(180); //buang
    delay(2000);
    Servo_Kapasitif.write(90); //kembalikan posisi
}

```

Gambar 4. 19 Memeriksa Kondisi Sensor Pemilah Sampah Organik

Kode ini memeriksa kondisi sensor setelah 1 detik dengan *delay*(1000). Jika sensor *Optical* 2 (dua) tidak mendeteksi objek (*Value_Optical_2* == 0) dan sensor Kapasitif mendeteksi material non-logam (*Value_Kapasitif* == 1), maka program akan mencetak informasi ke konsol serial untuk menunjukkan bahwa sampah yang terdeteksi adalah organik. Setelah itu, motor *servo* yang terhubung ke *Servo_Kapasitif* akan diputar ke posisi 180° untuk membuang sampah ke tong organik. Setelah 2 (dua) detik, *servo* dikembalikan ke posisi 90° untuk posisi awal.

```

if (Value_Kapasitif == 0 && Value_Optical_2 == 0) {
    delay(1000);
    //cek console untuk deteksi jenis sampah
    Serial.print("ANORGANIK ==> ");
    Serial.print(Value_Kapasitif);
    Serial.print(" OPTIK ==>");
    Serial.println(Value_Optical_2);

    /*putar servo ke tong organik*/
    Servo_Kapasitif.write(0); //buang
    delay(2000);
    Servo_Kapasitif.write(90); //kembalikan posisi
}

```

Gambar 4. 20 Memeriksa Kondisi Sensor Pemilah Sampah Anorganik

Kode ini memeriksa kondisi sensor setelah 1 (satu) detik dengan *delay*(1000). Jika sensor Kapasitif tidak mendeteksi material non-logam

(*Value_Kapasitif* == 0) dan sensor *Optical* 2 (dua) juga tidak mendeteksi objek (*Value_Optical_2* == 0), maka program akan mencetak informasi ke konsol serial untuk menunjukkan bahwa sampah yang terdeteksi adalah anorganik. Setelah itu, motor *servo* yang terhubung ke *Servo_Kapasitif* akan diputar ke posisi 0 derajat untuk membuang sampah ke tong anorganik. Setelah 2 (dua) detik, *servo* dikembalikan ke posisi 90 derajat untuk posisi awal.

```
digitalWrite(Value_Alarm, 0);
if ( Value_Alarm == 1 && Value_Kapasitif == 1 && Value_Optical_2 == 0){
    digitalWrite(Alarm, 1);
    delay(100);
    digitalWrite(Alarm, 0);
    Servo_Kapasitif.write(0);//buang
    delay(200);
    Servo_Kapasitif.write(90);
```

Gambar 4. 21 Mengaktifkan Alarm dan Menggerakkan *Servo*

Kode ini memeriksa kondisi untuk mengaktifkan alarm dan menggerakkan *servo*. Pertama, *digitalWrite(Value_Alarm, 0)* memastikan alarm dalam keadaan mati. Jika *Value_Alarm* == 1, *Value_Kapasitif* == 1, dan *Value_Optical_2* == 0, yang berarti kondisi tertentu telah terpenuhi, maka alarm diaktifkan dengan *digitalWrite(Alarm, 1)*. Setelah 100 (seratus) milidetik, alarm dimatikan lagi dengan *digitalWrite(Alarm, 0)*. Kemudian, motor *servo* yang terhubung ke *Servo_Kapasitif* diputar ke posisi 0 derajat untuk membuang sampah, lalu setelah 200 milidetik, *servo* dikembalikan ke posisi 90 derajat untuk posisi awal.

```
kapasitas_sampah
#include <Servo.h>
#include <FirebaseESP8266.h>
#include <ESP8266WiFi.h>
```

Gambar 4. 22 Meyertakan Pustaka (*Library*)

Perintah ini digunakan untuk menyertakan pustaka (*Library*) *Servo* yang memungkinkan untuk mengontrol motor *servo* menggunakan mikrokontroler seperti ESP8266. Dan `#include <FirebaseESP8266.H>` Digunakan untuk berinteraksi dengan *Firebase* menggunakan ESP8266. `#include <ESP8266WiFi.h>` digunakan untuk menghubungkan ESP8266 ke jaringan *WiFi*.

```
#define WIFI_SSID "Trashbin"
#define WIFI_PASSWORD "qwerty123"
int wifistatus;

FirebaseData fbdo;
```

Gambar 4. 23 Menghubungkan Ke Jaringan *Wifi*,

Perintah di atas digunakan untuk menghubungkan ke jaringan *WiFi*, dan `int wifistatus`, `FirebaseData fbdo` digunakan untuk mendeklarasikan objek untuk status *Wifi* dan *Firebase*.

```
#define FIREBASE_HOST "https://trashbinmonitor-53f39-default-rtdb.firebaseio.com/"
#define FIREBASE_AUTH "dTLo9yezIwqEbBtAuP4T2WkeLJ43qG1eEOs9up8W"
```

Gambar 4. 24 Mendefinisikan Alamat dan *Token Autentikasi*

Kode ini digunakan untuk mendefinisikan alamat dan token autentikasi yang diperlukan untuk menghubungkan aplikasi dengan *Firebase*. `FIREBASE_HOST` menyimpan URL dari *Firebase Realtime Database*, yang digunakan untuk mengakses dan menyimpan data. `FIREBASE_AUTH` berisi token *Autentikasi* yang memastikan bahwa aplikasi Anda memiliki izin untuk berinteraksi dengan *database* tersebut. Keduanya sangat penting untuk memastikan aplikasi dapat terhubung dengan benar ke *Firebase* dan mengelola data di dalamnya.

```

int trig1 = D4; //variable sensor ultrasonik 1 volume sampah logam
int echo1 = D5; //variable sensor ultrasonik 1 volume sampah logam

int trig2 = D7; //variable sensor ultrasonik 2 volume sampah anorganik
int echo2 = D6; //variable sensor ultrasonik 2 volume sampah anorganik

int trig3 = D0; //variable sensor ultrasonik 3 volume sampah Organik
int echo3 = D1; //variable sensor ultrasonik 3 volume sampah Organik

```

Gambar 4. 25 Mendeklarasikan Beberapa Variabel

Kode ini mendeklarasikan beberapa variabel untuk menghubungkan sensor ultrasonik yang digunakan untuk mengukur volume sampah berdasarkan jenisnya. *trig1* dan *echo1* masing-masing adalah *pin* yang digunakan untuk sensor ultrasonik pertama yang mengukur volume sampah logam. Begitu pula, *trig2* dan *echo2* digunakan untuk sensor ultrasonik kedua yang mengukur volume sampah anorganik. Terakhir, *trig3* dan *echo3* adalah *pin* yang terhubung dengan sensor ultrasonik ketiga untuk mengukur volume sampah organik. Setiap pasangan *pin trig* dan *echo* bekerja untuk mengirim dan menerima gelombang ultrasonik, yang digunakan untuk mengukur jarak atau volume sampah.

```

void setup() {
  Serial.begin(9600);

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD); //PanggilWiFi
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.println(".");
    delay(500);
  }
  Serial.println();
  Serial.print("connected : "); //tampil hasil
  Serial.println(WiFi.localIP()); //tampil alamat ip

  Firebase.begin (FIREBASE_HOST, FIREBASE_AUTH); //Mulai koneksi ke firebase

```

Gambar 4. 26 Mengatur Koneksi *Wi-Fi* dan *Firebase*

Kode ini digunakan untuk mengatur koneksi *Wi-Fi* dan *Firebase* saat program dimulai. Fungsi *Serial.begin(9600)* memulai komunikasi serial dengan

kecepatan 9600 *baud*. Kemudian, *WiFi.begin(WIFI_SSID, WIFI_PASSWORD)* mencoba menghubungkan perangkat ke jaringan *Wi-Fi* menggunakan SSID dan kata sandi yang telah ditentukan. Selama proses koneksi, program akan mencetak titik (".") di serial monitor setiap 500 *milidetik* untuk menunjukkan bahwa koneksi masih berlangsung. Setelah terhubung, alamat IP perangkat akan ditampilkan di serial monitor. Terakhir, *Firestore.begin(FIREBASE_HOST, FIREBASE_AUTH)* memulai koneksi ke *Firestore* menggunakan alamat *host* dan *token autentikasi* yang telah didefinisikan sebelumnya.

```
pinMode(trig1, OUTPUT); // inisialisais sensor ultrasonik 1 volume sampah Logam
pinMode(echo1, INPUT); // inisialisais sensor ultrasonik 1 volume sampah Logam

pinMode(trig2, OUTPUT); // inisialisais sensor ultrasonik 2 volume sampah Nonlogam
pinMode(echo2, INPUT); // inisialisais sensor ultrasonik 2 volume sampah Nonlogam

pinMode(trig3, OUTPUT); // inisialisais sensor ultrasonik 3 volume sampah Organik
pinMode(echo3, INPUT); // inisialisais sensor ultrasonik 3 volume sampah Organik
```

Gambar 4. 27 Mengatur Pin-Pin yang Terhubung Dengan Sensor Ultrasonik

Kode ini digunakan untuk mengatur *pin-pin* yang terhubung dengan sensor ultrasonik untuk mengukur volume sampah. Dengan menggunakan *pinMode()*, setiap *pin* dikonfigurasi sebagai *OUTPUT* atau *INPUT*. *Pin trig1* dan *echo1* di *inisialisasi* untuk sensor ultrasonik pertama yang digunakan untuk mengukur volume sampah logam, di mana *trig1* akan mengirimkan sinyal dan *echo1* menerima sinyal pantul. Begitu pula, *trig2* dan *echo2* digunakan untuk sensor ultrasonik kedua yang mengukur volume sampah non-logam, dan *trig3* serta *echo3* digunakan untuk sensor ultrasonik ketiga yang mengukur volume sampah organik. Setiap sensor memiliki *pin* yang sesuai untuk mengirim dan menerima sinyal.

```

void loop() {
  //Variabel logam
  long duration1, logam;
  int kapasitasLogam;

  //mengirim sinyal ultrasonik
  digitalWrite(trig1, LOW);
  delayMicroseconds(8);
  digitalWrite(trig1, HIGH);
  delayMicroseconds(8);
  digitalWrite(trig1, LOW);
  delayMicroseconds(8);
  duration1 = pulseIn(echo1, HIGH); //menerima echo
  logam = (duration1 / 2) / 29.1; //menghitung jarak berdasarkan waktu echo

  if (logam <= 7) { //Jika nilai logam lebih kecil sama dengan 7 maka hasilnya 100%
    Serial.println("100% Logam");
    kapasitasLogam = 100;
  } else
  if (logam >=8 && logam <= 12) { // Jika nilai logam lebih besar sama dengan 6 dan logam lebih kecil sama dengan 12 maka hasilnya 75%
    Serial.println("75% Logam");
    kapasitasLogam=75;
  } else
  if (logam >=13 && logam <=17) { //Jika nilai logam lebih besar sama dengan 13 dan logam lebih kecil sama dengan 17 maka hasilnya 50%
    Serial.println("50% Logam");
    kapasitasLogam=50;
  }else
  if (logam >=16 && logam <=20) { //Jika nilai logam lebih besar sama dengan 16 dan logam lebih kecil sama dengan 20 maka hasilnya 25%
    Serial.println("25% Logam");
    kapasitasLogam=25;
  } else if (logam >= 21 && logam <= 24) { //jika logam lebih besar sama dengan 21 dan logam lebih kecil sama dengan 24 maka hasilnya 0%
    Serial.println("0% Logam");
    kapasitasLogam=0;
  }
  delay(200);
  Firebase.setInt(fbdo, "logam", kapasitasLogam); // mengirim data jarak ke Firebase
  delay(500);
}

```

Gambar 4. 28 Mengukur Jumlah Sampah Logam

Kode ini mengukur jumlah sampah logam menggunakan sensor ultrasonik. Program mengirimkan sinyal ultrasonik melalui *pin trig1* dan kemudian mengukur waktu yang dibutuhkan sinyal untuk kembali dengan menggunakan *pulseIn()* pada *pin echo1*. Berdasarkan waktu yang dihitung, program mengonversi waktu tersebut menjadi jarak dan menentukan kapasitas sampah logam dalam lima kategori: 100%, 75%, 50%, 25%, atau 0%. Program ini mencetak hasilnya ke serial monitor dan mengirimkan data kapasitas logam ke *Firestore* untuk disimpan menggunakan *Firestore.setInt()*. Program terus melakukan pengukuran dan pengiriman data setiap 500 milidetik.

```

//Variable anorganik
long duration2, anorganik;
int kapasitasAnorganik;
//mengirim sinyal ultrasonik
digitalWrite(trig2, LOW);
delayMicroseconds(8);
digitalWrite(trig2, HIGH);
delayMicroseconds(8);
digitalWrite(trig2, LOW);
delayMicroseconds(8);

duration2 = pulseIn(echo2, HIGH); //menerima echo
anorganik = (duration2 / 2) / 29.1; //menghitung jarak berdasarkan waktu echo
if (anorganik <= 7) { //Jika nilai anorganik lebih kecil sama dengan 7 maka hasilnya 100%
    Serial.println("100% Anorganik");
    kapasitasAnorganik = 100;
} else
    if (anorganik >=8 && anorganik <= 12) { // Jika nilai anorganik lebih besar sama dengan 6 dan anorganik lebih kecil sama dengan 12 maka hasilnya 75%
        Serial.println("75% Anorganik");
        kapasitasAnorganik=75;
    } else
        if (anorganik >=13 && anorganik <=17) { //Jika nilai anorganik lebih besar sama dengan 13 dan anorganik lebih kecil sama dengan 17 maka hasilnya 50%
            Serial.println("50% Anorganik");
            kapasitasAnorganik=50;
        } else
            if (anorganik >=16 && anorganik <=20) { //Jika nilai anorganik lebih besar sama dengan 16 dan anorganik lebih kecil sama dengan 20 maka hasilnya 25%
                Serial.println("25% Anorganik");
                kapasitasAnorganik=25;
            } else if (anorganik >= 21 && anorganik <= 24) { //jika anorganik lebih besar sama dengan 21 dan anorganik lebih kecil sama dengan 24 maka hasilnya 0%
                Serial.println("0% Anorganik");
                kapasitasAnorganik=0;
            }
}
delay(100);
Firebase.setInt(fbdo, "/anorganik", kapasitasAnorganik); // mengirim data jarak ke Firebase
delay(100);

```

Gambar 4. 29 Mengukur Volume Sampah Anorganik

Kode ini digunakan untuk mengukur volume sampah anorganik menggunakan sensor ultrasonik. Pertama, sinyal dikirimkan melalui *pin trig2*, kemudian program mengukur waktu yang dibutuhkan untuk sinyal kembali dengan menggunakan *pulseIn()* pada *pin echo2*. Berdasarkan waktu yang diterima, jarak dihitung dan dikonversi menjadi nilai kapasitas sampah anorganik dalam lima kategori: 100%, 75%, 50%, 25%, atau 0%. Hasil pengukuran kapasitas sampah anorganik ini dicetak ke serial monitor dan kemudian dikirim ke *Firestore* untuk disimpan menggunakan *Firestore.setInt()*. Program ini terus mengirimkan data setiap 100 milidetik.

```

//organik
long duration3, organik;
int kapasitasOrganik;
digitalWrite(trig3, LOW);
delayMicroseconds(8);
digitalWrite(trig3, HIGH);
delayMicroseconds(8);
digitalWrite(trig3, LOW);
delayMicroseconds(8);
duration3 = pulseIn(echo3, HIGH);
organik = (duration3 / 2) / 29.1;

if (organik <= 7) { //Jika nilai organik lebih kecil sama dengan 7 maka hasilnya 100%
  Serial.println("100% Organik");
  kapasitasOrganik = 100;
} else
if (organik >=8 && organik <= 12) { // Jika nilai organik lebih besar sama dengan 6 dan organik lebih kecil sama dengan 12 maka hasilnya 75%
  Serial.println("75% Organik");
  kapasitasOrganik=75;
} else
if (organik >=13 && organik <=17) { //Jika nilai organik lebih besar sama dengan 13 dan organik lebih kecil sama dengan 17 maka hasilnya 50%
  Serial.println("50% Organik");
  kapasitasOrganik=50;
} else
if (organik >=16 && organik <=20) { //Jika nilai organik lebih besar sama dengan 16 dan organik lebih kecil sama dengan 20 maka hasilnya 25%
  Serial.println("25% Organik");
  kapasitasOrganik=25;
} else if (organik >= 21 && organik <= 24) { //jika organik lebih besar sama dengan 21 dan organik lebih kecil sama dengan 24 maka hasilnya 0%
  Serial.println("0% Organik");
  kapasitasOrganik=0;
}
}

delay(100);
Firebase.setInt(fbdo, "/organik", kapasitasOrganik); // mengirim data jarak ke Firebase
delay(100);
}

```

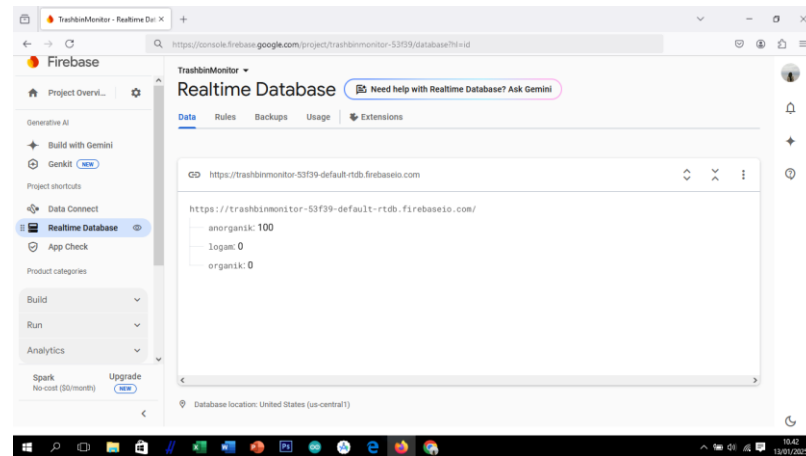
Gambar 4. 30 Mengukur Volume Sampah Organik

Kode ini digunakan untuk mengukur volume sampah organik menggunakan sensor ultrasonik. Program pertama-tama mengirimkan sinyal melalui *pin trig3* dan kemudian mengukur waktu yang diperlukan untuk sinyal kembali menggunakan *pulseIn()* pada *pin echo3*. Berdasarkan waktu yang dihitung, program menentukan kapasitas sampah organik dalam lima kategori, yaitu 100%, 75%, 50%, 25%, atau 0%. Hasil pengukuran kapasitas organik ini ditampilkan di serial monitor dan kemudian dikirimkan ke *Firestore* menggunakan *Firestore.setInt()* untuk disimpan. Program mengirimkan data setiap 100 milidetik.

a. *Realtime Database*

Merupakan tampilan data *Realtime Database* dikirimkan dengan SDK seluler dan web sehingga dapat mem-*build* aplikasi tanpa memerlukan server. juga dapat mengeksekusi kode *backend* yang merespons peristiwa yang dipicu oleh

database menggunakan *Cloud Functions for Firebase*. Tampilan pada gambar dibawa merupakan kode *database* yang terhubung ke aplikasi android, terdapat1(satu) Tabel yang berisi Anorganik dengan *value* 100, Logam *value* 0 dan Organik *value* 0.



Gambar 4. 31 Tampilan *Database Realtime Firebase*

```
implementation 'androidx.appcompat:appcompat:1.6.1'
implementation 'com.google.android.material:material:1.5.0'
implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
implementation 'com.google.firebase:firebase-database:20.1.0'
implementation 'androidx.core:core-ktx:latest_version'
implementation 'com.google.firebase:firebase-messaging'
//implementation 'com.android.support:support-compat:28.0.0'
testImplementation 'junit:junit:4.13.2'
androidTestImplementation 'androidx.test.ext:junit:1.1.5'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
implementation platform('com.google.firebase:firebase-bom:32.0.0') // Firebase BOM
implementation 'com.google.firebase:firebase-analytics' // Tambahkan Layanan Firebase (contoh: Analytics)
```

Gambar 4. 32 File Konfigurasi Gradle Di Proyek Android Untuk Mendefinisikan Pustaka (Library)

Kode tersebut adalah bagian dari *file* konfigurasi *Gradle* di *proyek Android* yang digunakan untuk mendefinisikan pustaka (*library*) yang diperlukan oleh aplikasi. Pustaka ini mencakup berbagai fungsi, seperti kompatibilitas antarmuka pengguna melalui *androidx.appcompat*, mendukung komponen desain material dengan *com.google.android.material*, dan tata letak fleksibel menggunakan *ConstraintLayout*. Selain itu, terdapat integrasi dengan *Firebase* melalui *firebase-database* untuk menggunakan *Realtime Database*, serta *firebase-messaging* untuk

mengelola notifikasi. Pustaka *Firebase* menggunakan BOM (*Bill of Materials*) agar versi setiap layanan *Firebase* saling kompatibel. Pustaka pengujian, seperti *JUnit* dan *Espresso*, juga ditambahkan untuk membantu pengembang menulis dan menjalankan tes aplikasi. Semua pustaka ini diatur agar otomatis diunduh dan digunakan, sehingga mempermudah proses pengembangan aplikasi Android.

b. Rancangan Aplikasi

1. Tampilan Aplikasi *Monitoring*

Tampilan ini merupakan tampilan keadaan tempat sampah Logam terisi 50% atau status penuh, tempat sampah Non Logam terisi 100% atau status terisi dan tempat sampah organik 50% status penuh sensor menunjukkan keadaan tempat sampah menyalah dan sensor mendeteksi tempat sampah kosong dari ketinggian 25cm. Pada aplikasi ini menampilkan jenis tempat sampah dan kapasitas sampah.



Gambar 4. 33 Tampilan Tempat Sampah Terisi dan Penuh

```

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/splogam"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="%"
        android:textAlignment="center"
        android:textSize="25dp"
        android:textStyle="bold" />
</LinearLayout>

```

Gambar 4. 34 Pengguna (*Ui Layout*) Dalam Aplikasi Android Untuk Tampilan Kapasitas Sampah Logam

Kode tersebut adalah bagian dari tata letak antarmuka pengguna (*UI layout*) dalam aplikasi Android yang menggunakan *LinearLayout* sebagai wadah (*container*). Di dalamnya terdapat sebuah elemen *TextView* dengan ID `@+id/splogam`, yang digunakan untuk menampilkan teks dengan simbol "%". Ukuran teks ditentukan dengan *textSize* sebesar 25dp, teks dirancang dengan gaya tebal (*bold*), dan posisinya diatur agar berada di tengah secara horizontal menggunakan *textAlignment="center"*. Tata letak ini memiliki lebar yang menyesuaikan layar perangkat (*match_parent*) dan tinggi yang sesuai dengan kontennya (*wrap_content*). Secara keseluruhan, elemen ini kemungkinan digunakan untuk menampilkan persentase, seperti kapasitas sampah logam dalam aplikasi.

```

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/spanorganik"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="%"
        android:textAlignment="center"
        android:textSize="25dp"
        android:textStyle="bold" />
</LinearLayout>

```

Gambar 4. 35 Pengguna (*Ui Layout*) Dalam Aplikasi Android Untuk Tampilan Kapasitas Sampah Anorganik

Kode tersebut adalah bagian dari tata letak antarmuka aplikasi Android yang menggunakan *LinearLayout* sebagai wadah untuk mengatur elemen di dalamnya. Elemen *TextView* dengan ID `@+id/sporganik` berfungsi untuk menampilkan teks dengan simbol "%". Properti *textSize* diatur sebesar 25dp untuk membuat teks terlihat lebih besar, dan teks diberi gaya tebal (*bold*) agar lebih menonjol. Teks ini juga disejajarkan di tengah secara horizontal menggunakan *textAlignment="center"*. Lebar tata letak disesuaikan dengan lebar layar perangkat (*match_parent*), sedangkan tingginya hanya menyesuaikan dengan isi (*wrap_content*). Elemen ini kemungkinan digunakan untuk menampilkan informasi persentase, seperti kapasitas sampah anorganik dalam aplikasi.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/sporganik"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAlignment="center"
        android:textStyle="bold"
        android:text="%"
        android:textSize="25dp" />
</LinearLayout>
```

Gambar 4. 36 Pengguna (*Ui Layout*) Dalam Aplikasi Android Untuk Tampilan Kapasitas Sampah Organik

Kode tersebut adalah bagian dari tata letak antarmuka aplikasi Android yang menggunakan *LinearLayout* sebagai wadah untuk mengatur elemen di dalamnya. Elemen *TextView* dengan ID `@+id/sporganik` berfungsi untuk menampilkan teks dengan simbol "%". Properti *textSize* diatur sebesar 25dp untuk membuat teks

terlihat lebih besar, dan teks diberi gaya tebal (*bold*) agar lebih menonjol. Teks ini juga disejajarkan di tengah secara horizontal menggunakan `textAlignment="center"`. Lebar tata letak disesuaikan dengan lebar layar perangkat (`match_parent`), sedangkan tingginya hanya menyesuaikan dengan isi (`wrap_content`). Elemen ini kemungkinan digunakan untuk menampilkan informasi persentase, seperti kapasitas sampah organik dalam aplikasi.

2. Tampilan Notifikasi Kapasitas Tempat sampah Penuh

Tampilan ini merupakan tampilan Notifikasi apabila keadaan tempat sampah Logam penuh atau 100% notifikasi akan muncul ketika sensor mendeteksi ketinggian sampah 100%.



Gambar 4. 37 Tampilan Notifikasi Tempat Sampah Penuh

```

DataLogam.addValueEventListener(new ValueEventListener() {
    2 usages
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        Integer data = dataSnapshot.getValue(Integer.class);
        if (data != null && data == 100) {
            showNotification("Sampah Logam Sudah Penuh");
        }
        logam.setText(data != null ? data + "%" : "");
    }
}

```

Gambar 4. 38 Perubahan Data Dari Data Logam dan Menampilkan Notifikasi

Kode di atas berfungsi untuk memantau perubahan data dari *Data Logam* dan menampilkan notifikasi jika data tersebut mencapai nilai tertentu. Pertama, fungsi *createNotificationChannel()* dipanggil untuk menyiapkan saluran notifikasi. Kemudian, *addValueEventListener* digunakan untuk memantau perubahan data pada *DataLogam*. Ketika data berubah, fungsi *onDataChange* dipanggil, yang mengambil nilai data dalam bentuk Integer. Jika nilai data tersebut adalah 100, maka aplikasi akan menampilkan notifikasi dengan pesan "Sampah Logam Sudah Penuh". Selain itu, teks pada elemen logam akan diperbarui sesuai dengan nilai data yang diterima, menampilkan persentase dari data tersebut.

```

DataAnorganik.addValueEventListener(new ValueEventListener() {
    2 usages
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        Integer data = dataSnapshot.getValue(Integer.class);
        if (data != null && data == 100) {
            showNotification("Sampah Anorganik Sudah Penuh");
        }
        anorganik.setText(data != null ? data + "%" : "");
    }
}

```

Gambar 4. 39 Perubahan Data Dari Anorganik dan Menampilkan Notifikasi

Kode di atas berfungsi untuk memantau perubahan data pada *DataAnorganik* dan menampilkan notifikasi jika data tersebut mencapai nilai tertentu. Fungsi *addValueEventListener* digunakan untuk mendengarkan perubahan data pada *DataAnorganik*. Ketika data berubah, metode *onDataChange* akan dipanggil, yang mengambil nilai data dalam bentuk Integer. Jika nilai data tersebut adalah 100, aplikasi akan menampilkan notifikasi dengan pesan "Sampah Anorganik Sudah Penuh". Selain itu, teks pada elemen anorganik akan diperbarui untuk menampilkan persentase nilai data yang diterima. Jika data tidak tersedia, maka teks akan kosong.

```

DataOrganik.addValueEventListener(new ValueEventListener() {
    2 usages
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        Integer data = dataSnapshot.getValue(Integer.class);
        if (data != null && data == 100) {
            showNotification("Sampah Organik Sudah Penuh");
        }
        organik.setText(data != null ? data + "%" : "");
    }
}

```

Gambar 4. 40 Perubahan Data Dari Data Organik dan Menampilkan Notifikasi

Kode di atas digunakan untuk memantau perubahan data pada *DataOrganik* dan memberikan notifikasi jika data tersebut mencapai angka tertentu. Dengan menggunakan *addValueEventListener*, aplikasi mendengarkan setiap perubahan pada *DataOrganik*. Ketika data berubah, metode *onDataChange* dipanggil untuk mengambil nilai data dalam bentuk Integer. Jika nilai data tersebut adalah 100, maka aplikasi akan menampilkan notifikasi dengan pesan "Sampah Organik Sudah Penuh". Selain itu, teks pada elemen organik akan diperbarui untuk menampilkan nilai data dalam bentuk persentase. Jika data tidak tersedia, teks akan kosong.

G. Pembahasan

1. Metode Pengujian

Metode pengujian sistem merupakan proses eksekusi suatu program atau sistem secara menyeluruh. Pengujian ini bertujuan untuk memastikan bahwa fungsi perangkat keras dan perangkat lunak beroperasi sesuai dengan spesifikasi yang ditetapkan, serta memenuhi persyaratan kinerja yang telah ditentukan.

2. Proses Mendeteksi Sampah

a. Sampah Logam

Proses mendeteksi jenis sampah logam menggunakan sensor *proximity* induktif yang mendeteksi keberadaan logam saat sampah melewati jalur tempat penyortiran. Jika sensor mendeteksi logam, mikrokontroler Arduino uno akan mengaktifkan motor *servo* untuk menggerakkan lengan mekanis atau pintu selektor ke posisi 0 (Nol) derajat, sehingga sampah logam dapat diarahkan ke tempat pembuangan khusus logam.

b. Sampah Anorganik

Proses mendeteksi jenis sampah anorganik menggunakan sensor *infrared* dan motor *servo* dimulai dengan sensor *infrared* (IR) yang mendeteksi pantulan cahaya dari objek sampah saat melewati jalur deteksi. Sampah anorganik seperti plastik, kaca memiliki karakteristik pantulan cahaya yang berbeda dibandingkan sampah organik. Mikrokontroler seperti Arduino memproses data dari sensor IR dan menentukan apakah sampah termasuk anorganik. Jika terdeteksi sebagai sampah anorganik, motor *servo* akan bergerak ke posisi 0 (Nol) derajat untuk mengarahkan sampah ke tempat pembuangan khusus.

c. Sampah Organik

Proses mendeteksi jenis sampah organik menggunakan sensor *infrared* dan sensor kapasitif dimulai dengan sensor *infrared* (IR) yang mendeteksi pantulan cahaya dari sampah, sementara sensor kapasitif mengukur perubahan kapasitansi untuk membedakan antara bahan organik dan anorganik berdasarkan sifat dielektriknya. Sampah organik seperti sisa makanan dan dedaunan umumnya memiliki kandungan air yang lebih tinggi, sehingga memberikan respons kapasitif yang khas. *Mikrokontroler* seperti *Arduino* memproses data dari kedua sensor untuk mengidentifikasi sampah organik. Jika terdeteksi sebagai organik, motor *servo* akan bergerak ke posisi 180° untuk mengarahkan sampah ke tempat pembuangan khusus Organik.

3. Pengujian Pemilah sampah 1

Pengujian Sensor Induktif dan Sensor *Infrared Proximity* Pada bagian pemilah 1 (satu) untuk sampah logam dan pemilah dua, dapat dilihat pada tabel di bawah ini:

Tabel 4. 8 Pengujian Pemilah 1

Objek	<i>Sensor Induktif Proximity</i>	<i>Sensor Infrared Proximity</i>	Motor Servo	Keterangan
Potongan Kayu	0	0	180°	Non Logam
Kertas	0	0	180°	Non Logam
Kaleng Minuman	1	0	0°	Logam
Botol Plastik	0	0	180°	Non Logam
Daun	0	0	180°	Non Logam

Plastik	0	0	180°	Non Logam
---------	---	---	------	-----------

Nilai awal Sensor Induktif *Proximity* adalah 0 (nol), yang berarti sensor ini tidak mendeteksi sampah. Sementara itu, nilai awal Sensor *Infrared Proximity* adalah 1 (satu), yang juga berarti sensor ini tidak mendeteksi.

Jika Sensor Induktif bernilai 1 (satu) dan Sensor *Infrared Proximity* bernilai 0 (nol), maka Motor *Servo* akan berputar 0° atau ke kiri, dan ini menandakan bahwa deteksinya adalah sampah logam. Sebaliknya, jika Sensor Induktif bernilai 0 (nol) dan Sensor *Infrared Proximity* juga bernilai 0 (nol), maka Motor *Servo* akan berputar 180° atau ke kanan, yang menunjukkan bahwa deteksinya adalah sampah non-logam.

4. Pengujian Pemilah sampah 2

Pengujian Sensor *Capacitive Proximity* dan sensor *Infrared Proximity* Pada bagian Pemilah 2 (dua) untuk sampah Non Logam dan sampah Organik, dapat dilihat pada Tabel dibawah ini :

Tabel 4. 9 Pengujian Pemilah 2

Objek	<i>Sensor Capacitive Proximity</i>	<i>Sensor Infrared Proximity</i>	Motor Servo	Keterangan
Potongan Kayu	1	0	180°	Organik
Kertas	0	0	0°	Anorganik
Plastik	0	0	0°	Anorganik
Botol Plastik	0	0	0°	Anorganik
Sayur	1	0	180°	Organik

Nilai awal Sensor *Capacitive Proximity* adalah 0 (nol), yang berarti sensor ini tidak mendeteksi. Sementara itu, nilai awal Sensor *Infrared Proximity* adalah 1 (satu), yang juga berarti sensor ini tidak mendeteksi.

Jika Sensor *Capacitive Proximity* bernilai 0 (nol) dan Sensor *Infrared Proximity* 0 (nol) maka motor *Servo* berputar 0° atau ke kiri yang mendeteksi sampah anorganik, jika sensor *Capacitive Proximity* bernilai 1 (satu) dan sensor *infrared* bernilai 0 (nol) maka motor *Servo* berputar 180° atau ke kanan yang menunjukkan bahwa mendeteksi sampah organik.

5. Pengujian sensor ultrasonik Kapasitas sampah

Tabel 4. 10 Pengujian *Sensor Ultrasonik* Kapasitas Sampah

Ketinggian	Kapasitas	Keterangan
24 - 21 cm	0%	Belum terisi
20 -16 cm	25%	Terisi
17 – 13 cm	50%	Terisi
12 – 8 cm	75%	Terisi
7 – 1 cm	100 %	Penuh

6. Pengujian *Black Box*

Tabel 4. 11 Pengujian *Black Box* Pemilah 1 (Satu) Sampah Logam

Uji coba	Hasil	Keterangan
Membuang sampah Logam	✓	Sampah logam terdeteksi, motor <i>servo</i> berputar 0° sampah berhasil terbang ke tong sampah logam
Foto dan <i>Screenshot</i>		



Tabel 4. 12 Pengujian *Black Box* Pemilah 1 (Satu) Sampah Non Logam

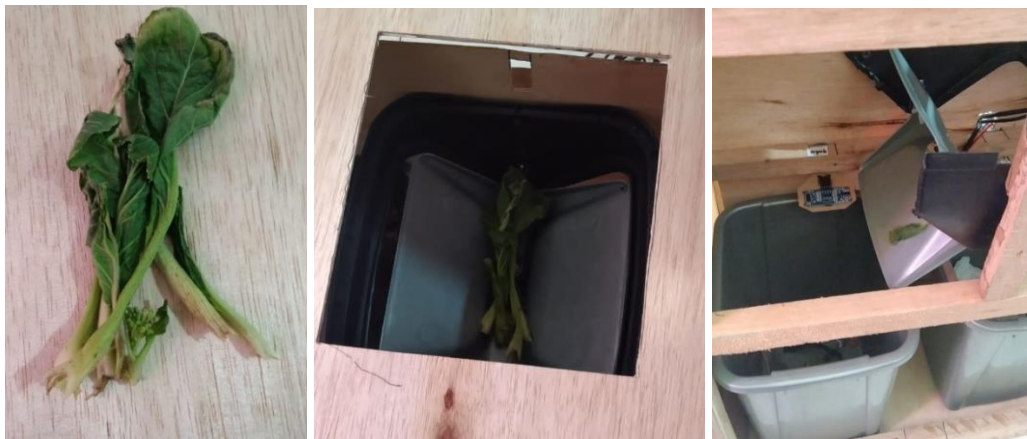
Uji coba	Hasil	Keterangan
Membuang sampah Non Logam	✓	Sampah logam tidak terdeteksi, motor <i>servo</i> berputar 180° sampah berhasil terbang ke Pemilah 2
Foto dan <i>Screenshot</i>		
		

Tabel 4. 13 Pengujian *Black Box* Pemilah 2 (Dua) Sampah Plastik

Uji coba	Hasil	Keterangan
Sampah Plastik terbang ke Pemilah 2	✓	Sampah Organik tidak terdeteksi, motor <i>servo</i> berputar 0° sampah

		berhasil terbang ke tong sampah anorganik
Foto dan <i>Screenshot</i>		
		

Tabel 4. 14 Pengujian *Black Box* Pemilah 2 (dua) Sampah Organik (daun)

Uji coba	Hasil	Keterangan
Sampah Daun terbang ke Pemilah 2	✓	Sampah Organik terdeteksi, motor servo berputar 180° sampah berhasil terbang ke tong sampah Organik
Foto dan <i>Screenshot</i>		
		

Tabel 4. 15 Keberhasilan Pengujian

No.	Nama Sampah	Jenis Sampah			Hasil
		Logam	Anorganik	Organik	
1	Kaleng Minuman	✓			Terdeteksi
2	Kertas		✓		Terdeteksi
3	Kantong Plastik		✓		Terdeteksi
4	Sayur			✓	Terdeteksi
5	Botol pelastik		✓		Terdeteksi
6	Kardus		✓		Terdeteksi

Pada **Tabel 4.15** Terdapat beberapa jenis sampah yang dapat terdeteksi oleh sensor pemilah sampah, seperti kaleng minuman terdeteksi sebagai jenis sampah logam, kertas, kantong plastik, botol plastik, serta kardus yang terdeteksi sebagai sampah anorganik. Sementara itu, sayur atau daun terdeteksi sebagai sampah organik. Namun, masih ada beberapa jenis sampah yang tidak terdeteksi oleh sensor pada prototipe pemilah sampah ini.

Tabel 4. 16 Tabel Pengujian Jenis Sampah Logam

Pengujian Sampah Logam Ke-	Terdeteksi	Tidak Terdeteksi	Keterangan
1	✓		Kaleng Minuman terdeteksi sensor <i>proximity</i> induktif dan <i>optical</i>
2	✓		Kaleng Minuman terdeteksi sensor <i>proximity</i> induktif dan <i>optical</i>
3	✓		Kaleng Minuman terdeteksi sensor <i>proximity</i> induktif dan <i>optical</i>
4	✓		Kaleng Minuman terdeteksi sensor <i>proximity</i> induktif dan <i>optical</i>
5	✓		Kaleng Minuman terdeteksi sensor <i>proximity</i> induktif dan <i>optical</i>
6	✓		Kaleng Minuman

			terdeteksi sensor <i>proximity</i> induktif dan <i>optical</i>
7	✓		Kaleng Minuman terdeteksi sensor <i>proximity</i> induktif dan <i>optical</i>
8		✓	Kaleng Minuman Tidak terdeteksi sensor <i>proximity</i> induktif dan hanya terdeteksi pada sensor <i>optical</i>
9	✓		Kaleng Minuman terdeteksi sensor <i>proximity</i> induktif dan <i>optical</i>
10	✓		Kaleng Minuman terdeteksi sensor <i>proximity</i> induktif dan <i>optical</i>

Pada **Tabel 4.16** pengujian jenis sampah logam tersebut dilakukan 10 kali pengujian pada sampah kaleng minuman, terdapat keberhasilan 9 kali pengujian berhasil pada Kaleng minuman terdeteksi sensor *proximity induktif* dan *optical* sehingga masuk ke wadah sampah logam dan 1 kali gagal adapun faktor yang menyebabkan gagal salah satunya ialah posisi sampah tidak tepat pada sensor Induktif sehingga pemilah pemutar kepemilah ke 2, adapun persentase keberhasilan dari 10 kali pengujian jenis sampah logam terdapat 90% keberhasilan.

Tabel 4. 17 Tabel Pengujian Jenis Sampah Anorganik

Pengujian Sampah Anorganik Ke-	Terdeteksi	Tidak Terdeteksi	Keterangan
1	✓		Sampah kertas terdeteksi sensor <i>optical</i>
2	✓		Sampah kertas terdeteksi sensor <i>optical</i>
3	✓		Sampah kertas terdeteksi sensor <i>optical</i>
4	✓		Sampah kertas terdeteksi sensor <i>optical</i>
5		✓	Sampah Kertas Tidak terdeteksi sensor <i>optical</i>

6	✓		Sampah kertas terdeteksi sensor <i>optical</i>
7	✓		Sampah kertas terdeteksi sensor <i>optical</i>
8	✓		Sampah kertas terdeteksi sensor <i>optical</i>
9	✓		Sampah kertas terdeteksi sensor <i>optical</i>
10	✓		Sampah kertas terdeteksi sensor <i>optical</i>

Pada **Tabel 4.17** pengujian jenis sampah anorganik tersebut dilakukan 10 (sepuluh) kali pengujian pada sampah kertas, terdapat keberhasilan 9 (sembilan) kali pengujian berhasil pada kertas terdeteksi sensor *optical* sehingga masuk ke wadah sampah anorganik dan 1 kali gagal adapun faktor yang menyebabkan gagal salah satunya ialah posisi sampah tidak tepat pada sensor *optical* sehingga masuk ke wadah organik, adapun persentase keberhasilan dari 10 (sepuluh) kali pengujian jenis sampah anorganik terdapat 90% keberhasilan.

Tabel 4. 18 Tabel Pengujian Jenis Sampah Organik

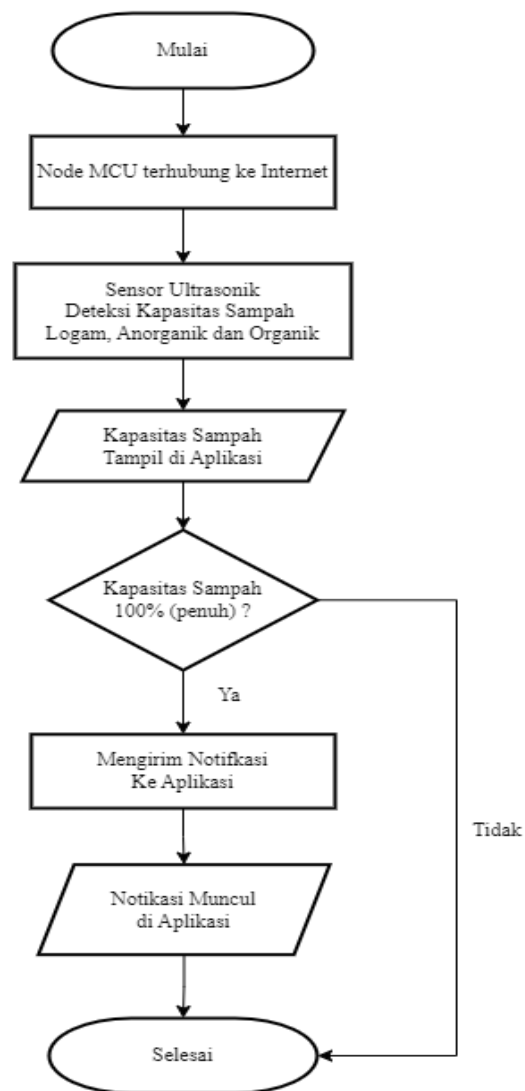
Pengujian Sampah Anorganik Ke-	Terdeteksi	Tidak Terdeteksi	Keterangan
1	✓		Sampah daun terdeteksi sensor <i>proximity</i> kapasitif dan <i>optical</i>
2	✓		Sampah daun terdeteksi sensor <i>proximity</i> kapasitif dan <i>optical</i>
3		✓	Sampah daun tidak terdeteksi sensor <i>proximity</i> kapasitif dan hanya terdeteksi sensor <i>optical</i>
4	✓		Sampah daun terdeteksi sensor <i>proximity</i> kapasitif dan <i>optical</i>
5	✓		Sampah daun terdeteksi sensor <i>proximity</i> kapasitif dan <i>optical</i>

6		✓	Sampah daun tidak terdeteksi sensor <i>proximity</i> kapasitif dan hanya terdeteksi sensor <i>optical</i>
7	✓		Sampah daun terdeteksi sensor <i>proximity</i> kapasitif dan <i>optical</i>
8	✓		Sampah daun terdeteksi sensor <i>proximity</i> kapasitif dan <i>optical</i>
9	✓		Sampah daun terdeteksi sensor <i>proximity</i> kapasitif dan <i>optical</i>
10	✓		Sampah daun terdeteksi sensor <i>proximity</i> kapasitif dan <i>optical</i>

Pada **Tabel 4.18** pengujian jenis sampah anorganik tersebut dilakukan 10 (sepuluh) kali pengujian pada sampah daun, terdapat keberhasilan 8 (delapan) kali pengujian berhasil pada Sampah daun terdeteksi sensor *proximity Kapasitif* dan *optical* sehingga masuk ke wadah sampah organik dan 2 (dua) kali gagal adapun faktor yang menyebabkan gagal salah satunya ialah posisi sampah tidak tepat pada sensor *proximity kapasitif* sehingga masuk ke wadah anorganik, adapun persentase keberhasilan dari 10 (sepuluh) kali pengujian jenis sampah organik terdapat 80% keberhasilan.

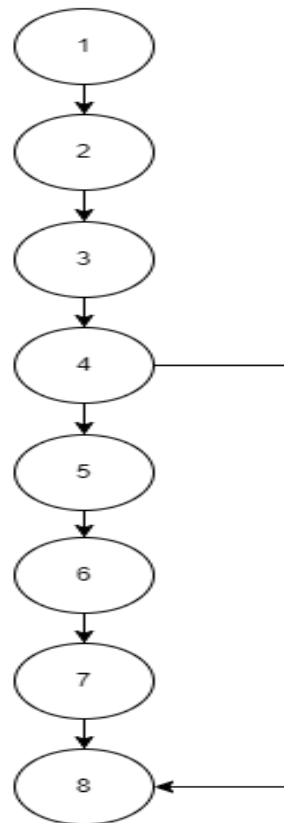
7. Pengujian *White Box*

a. *Flowchart* Aplikasi



Gambar 4. 41 *Flowchart* Notifikasi Aplikasi Monitoring Sampah

b. Flowgraph aplikasi



Gambar 4. 42 *Flowgraph* Sistem Aplikasi

Dari **Gambar 4.42**, *flowgraph* di atas dapat dilakukan proses perhitungan sebagai berikut:

1. Mengetahui *Cyclomatic Complexity* $V(G)$ dari *Edge* dan *Node*:

Dengan Rumus :

$$V(G) = E - N + 2$$

$$\text{Diketahui : } N (\text{Node}) = 8$$

$$E (\text{Edge}) = 7$$

$$P (\text{Predikat node}) = 1$$

$$\text{Penyelesaian : } V(G) = (E - N) + 2$$

$$= (7 - 8) + 2$$

$$= 1$$

$$\text{Perdikat Node (N)} = P + 1$$

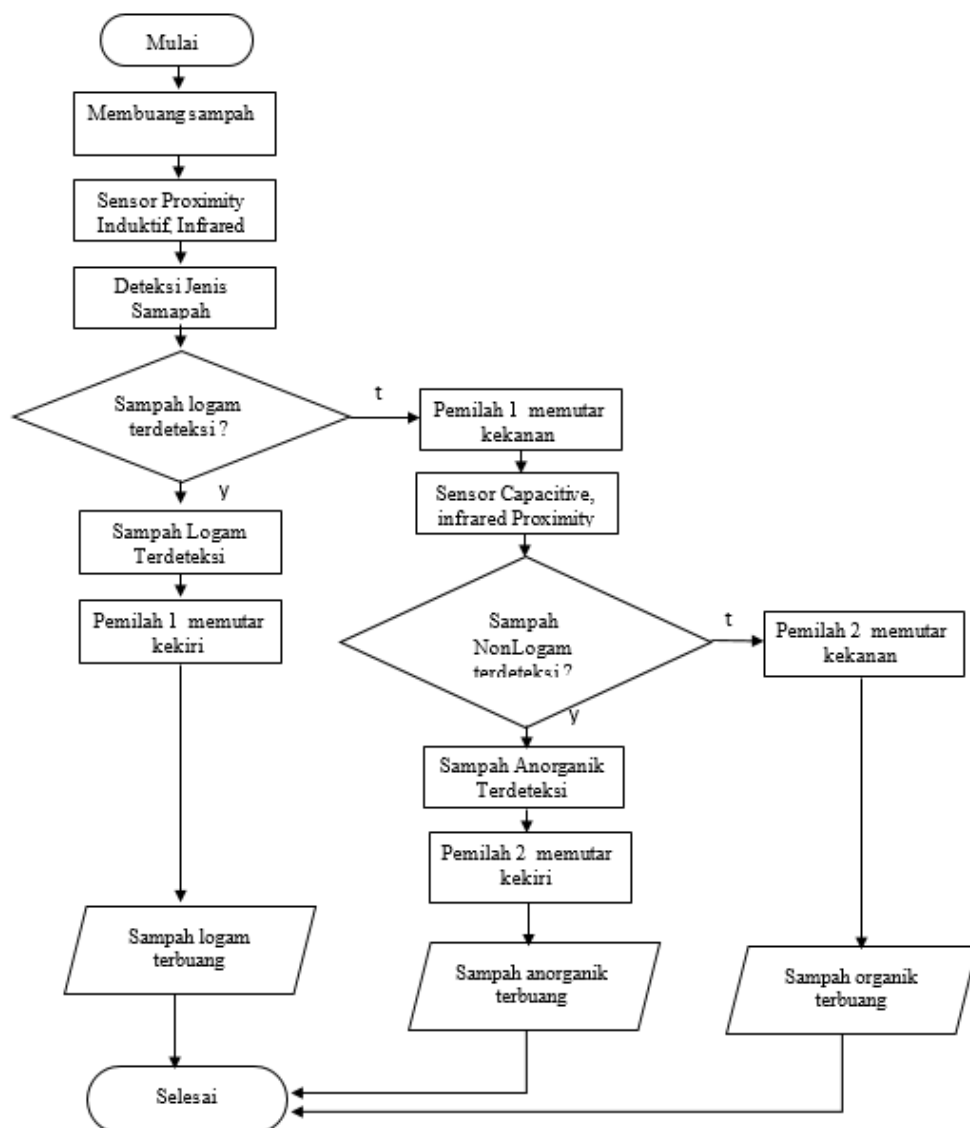
$$= 1 + 1$$

$$= 2$$

Path 1 : 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8

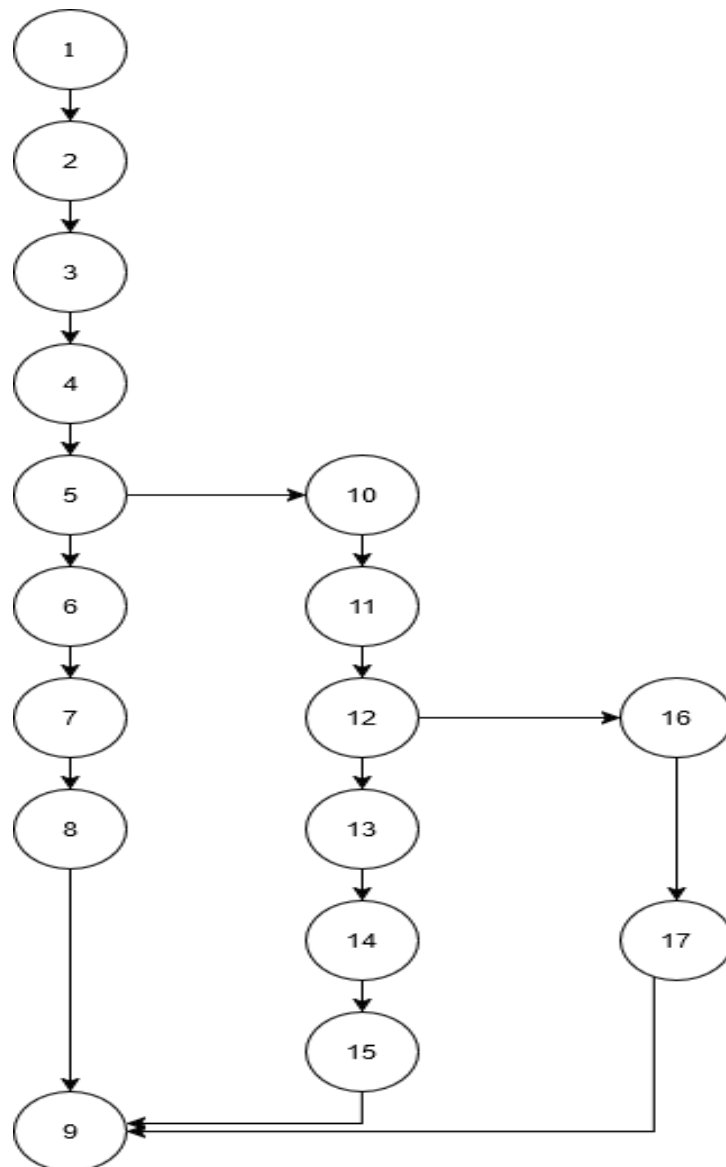
Path 2 : 1 – 2 – 3 – 4 – 8

c. Flowchart sistem pemilah Sampah Otomatis



Gambar 4. 43 Flowchart Sistem Pemilah Sampah Otomatis

d. Flowgraph Sistem Pemilah Sampah Otomatis



Gambar 4. 44 *Flowgraph* Sistem Pemilah Sampah Otomatis

Dari **Gambar 4.44**, *flowgraph* di atas dapat dilakukan proses perhitungan sebagai berikut:

1. Mengetahui *Cyclomatic Complexity* $V(G)$ dari *Edge* dan *Node*:

Dengan Rumus :

$$V(G) = E - N + 2$$

Diketahui : N (*Node*) = 17

E (*Edge*) = 18

$$P \text{ (Predikat node)} = 1$$

$$\begin{aligned} \text{Penyelesaian : } V(G) &= (E - N) + 2 \\ &= (18 - 17) + 2 \\ &= 3 \end{aligned}$$

$$\begin{aligned} \text{Predikat Node (N)} &= P + 1 \\ &= 2 + 1 \\ &= 3 \end{aligned}$$

$$\text{Path 1 : } 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9$$

$$\text{Path 2 : } 1 - 2 - 3 - 4 - 5 - 10 - 11 - 12 - 13 - 14 - 15 - 9$$

$$\text{Path 3 : } 1 - 2 - 3 - 4 - 5 - 10 - 11 - 12 - 16 - 17 - 9$$

BAB V

PENUTUP

A. Kesimpulan

Berdasarkan hasil penelitian dan pembahasan yang telah dijelaskan sebelumnya, maka sapat di tarik kesimpulan sebagai berikut :

1. Pada rancang bangun pemilah sampah ini digunakan Sensor *Proximity* Kapasitif, Sensor *Proximity* Induktif, dan Sensor *Proximity* Inframerah sebagai alat pendeteksi jenis sampah.
2. Sistem ini mampu melakukan pemilahan sampah berdasarkan jenisnya, yaitu logam, anorganik, dan organik.
3. Ketinggian sampah berdasarkan jenisnya dapat diukur menggunakan sensor ultrasonik.
4. Sampah logam akan dideteksi menggunakan sensor induktif *proximity* yang terpasang pada pemilah 1, sedangkan sampah organik dan organik dideteksi menggunakan sensor *capacitive proximity* dan sensor *optical* pada pemilah 2, keberhasilan pemilah jenis sampah logam dari 10 kali pengujian 9 kali berhasil 1 tidak berhasil sehingga hasil persentase keberhasilan 90%, pemilah jenis sampah anorganik dari 10 kali pengujian 9 kali berhasil 1 tidak berhasil sehingga hasil persentase keberhasilan 90% dan pemilah jenis sampah organik dari 10 kali pengujian 8 kali berhasil 2 tidak berhasil sehingga hasil persentase keberhasilan 80%.

5. Ketinggian sampah yang dideteksi mulai dari 24 - 21 cm atau kapasitas 0%, 20 – 16 cm berkapasitas 25%, 17 – 13 cm berkapasitas 50%, 12 – 8 cm berkapasitas 75% dan 7 – 1 cm berkapasitas 100%.
6. Kondisi kapasitas sampah 100% akan mengirim notifikasi ke aplikasi android sebagai informasi tempat sampah penuh. Pengiriman data melalui *Firestore* dengan *Realtime Database* dan NODEMCU ESP8266 .

B. Saran

Pada penelitian ini penulis merasa masih banyak kekurangan terhadap penelitian ini dan masih banyak hal yang bisa dikembangkan pada alat atau aplikasi *monitoring* berbasis android ini dengan hasil yang lebih akurat. Untuk peneliti selanjutnya saya harap dapat mengembangkan atau membuat alat dan aplikasi sejenisnya untuk mengoptimalkan alat pemilah sampah berbasis *IoT* dengan metode pengelolaan sampah konvensional yang ada. Dan bisa menampilkan *error* jika tidak murni dengan logam maupun non logam, hal ini akan memberikan pemahaman lebih jelas tentang manfaat dan keunggulan dari alat yang diusulkan.

Demikian saran yang penulis sampaikan, semoga saran yang penulis berikan dapat bermanfaat atau sebagai perbandingan untuk penelitian selanjutnya dan bagi para pembaca.

DAFTAR PUSTAKA

- Alfian, D. M., Sasmito, A. P., & Vendyansyah, N. (2021). IMPLEMENTASI LOGIKA FUZZY PADA RANCANG BANGUN SISTEM IRIGASI BERBASIS ARDUINO. In *Jurnal Mahasiswa Teknik Informatika*) (Vol. 5, Issue 1).
- Andrianto, M. (2019). PENERAPAN IOT PADA PERAWATAN TANAMAN DI DALAM RUMAH. In *Jurnal Mahasiswa Teknik Informatika* (Vol. 3, Issue 1).
- Endra, R. Y., Cucus, A., Afandi, F. N., & Syahputra, M. B. (2019). Model Smart Room Dengan Menggunakan Mikrokontroler Arduino Untuk Efisiensi Sumber Daya. *Explore: Jurnal Sistem Informasi Dan Telematika*, 10(1). <https://doi.org/10.36448/jsit.v10i1.1212>
- Fatmawati, K., Sabna, E., & Irawan, Y. (2020). Rancang Bangun Tempat Sampah Pintar Menggunakan Sensor Jarak Berbasis Mikrokontroler Arduino. *Riau Journal Of Computer Science*, 6(2), 124–134.
- indra, S., & Jasa, M. (2022). Rancang Bangun Tempat Sampah Pintar Pemilah Logam, Non Logam Dan Organik Otomatis Berbasis Iot (Internet of Things). *Science Electro*, nn, No. nn.
- Ibnu Hajar, M. H., & Jupri, S. (2021). Sistem Perancangan Tempat Sampah Logam dan Non Logam dengan menggunakan Aplikasi M.I.T Inventor. *Jurnal Teknologi Elektro*, 12(1), 35. <https://doi.org/10.22441/jte.2021.v12i1.007>
- Laksana, F. T., Nurhasanah, R., Wathoni, M., Informasi, P. T., Pendidikan, F. I., Muhammadiyah, U., & Selatan, K. T. (n.d.). RANCANG BANGUN SISTEM SMART TRASH BIN BERBASIS IoT MENGGUNAKAN ARDUINO UNO. 29–34.
- PRATAMA, A. S. (2018). PEMANFAATAN INTERNET OF THING (IoT) UNTUK SISTEM LAMPU DALAM RUANGAN. 3(3), 23–37.
- Rachmat Farhan, Muhaimin, & Maimun. (2019). Rancang Bangun Tempat Sampah Pintar Pada Gedung Jurusan Teknik Elektro Berbasis Mikrokontroler Arduino Mega 2560. *Jurnal Tektro*, 3(2), 119–124.
- Safiroh W.P, P. N., Nama, G. F., & Komarudin, M. (2022). Sistem Pengendalian Kadar PH dan Penyiraman Tanaman Hidroponik Model Wick System. *Jurnal Informatika Dan Teknik Elektro Terapan*, 10(1). <https://doi.org/10.23960/jitet.v10i1.2260>
- Wafi, A., Setyawan, H., & Ariyani, S. (2020). Prototipe Sistem Smart Trash Berbasis IOT (Internet Of Things) dengan Aplikasi Android. *Jurnal Teknik Elektro Dan Komputasi (ELKOM)*, 2(1), 20–29. <https://doi.org/10.32528/elkom.v2i1.3134>
- Yunus, M. (2018). Rancang Bangun Prototipe Tempat Sampah Pintar Pemilah Sampah Organik Dan Anorganik Menggunakan Arduino. *Proceeding STIMA*, 1(1), 340–343.

<https://firebase.google.com/products/realtime-database?hl=id>