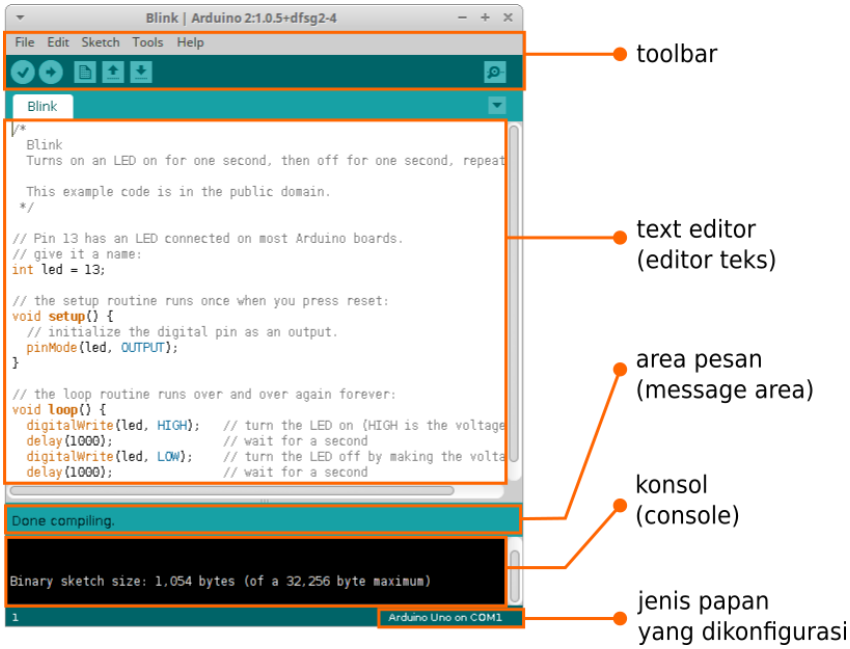
II.4 Teknologi Pendukung

Teknologi pendukung dibutuhkan untuk mendapatkan pemahaman yang baik mengenai teknologi yang digunakan pada sistem yang akan direalisasikan. Teknologi pendukung tersebut yang dikhususkan untuk sistem yang akan penulis rancang yaitu Arduino IDE, Google Speech API, dan App Inventor 2 (AI2).

II.4.1 Arduino IDE

Arduino Integrated Development Environment (IDE) adalah aplikasi lintas-platform (untuk Windows, macOS, Linux) yang ditulis dalam bahasa pemrograman Java. *Software* ini digunakan untuk menulis dan mengunggah program ke Arduino. Kode sumber untuk IDE dirilis di bawah GNU General Public License, versi 2. Arduino IDE mendukung bahasa C dan C ++ menggunakan aturan khusus penataan kode. Arduino IDE memasok perpustakaan perangkat lunak dari proyek pengkabelan yang menyediakan banyak prosedur input dan output umum. Kode yang ditulis pengguna hanya memerlukan dua fungsi dasar, untuk memulai sketsa dan *loop* program utama, yang dikompilasi dan dihubungkan dengan program stub main () ke dalam program eksekutif siklik yang dapat dieksekusi dengan GNU *toolchain*, juga disertakan dengan distribusi IDE. Arduino IDE menggunakan program avrdude untuk mengubah kode yang dapat dieksekusi menjadi file teks dalam pengkodean heksadesimal yang dimuat ke papan Arduino oleh program loader di firmware papan.



Gambar 2.2 Struktur Tampilan Arduino IDE

Secara umum, Arduino IDE terdiri atas beberapa bagian yang memiliki sub-sub fungsi tertentu. Kumpulan instruksi dapat ditemukan dalam opsi menu

berupa *File*, *Edit*, dan *Sketch*. Beberapa pengaturan dapat dilakukan melalui menu *Tools* serta bantuan yang dapat dijumpai dengan melakukan klik pada opsi menu *Help*.

1. Kumpulan Instruksi Umum

Instruksi	**Fungsi**
Verify	Memeriksa kode untuk kesalahan kompilasi.
Upload	Mengkompilasi kode dan mengunggahnya ke *platform* yang dikonfigurasi.
New	Membuat sketsa baru.
Open	Menampilkan sebuah menu dari semua sketsa pada *sketchbook*. Mengklik salah satunya akan membuka ke dalam jendela yang digunakan dan menggantikan isinya.
Save	Menyimpan sketsa.
Serial Monitor	Membuka serial monitor.
Upload Using Programmer	Menulis ulang *bootloader* pada *platform*.
Export Compiled Binary	Menyimpan sebuah berkas .hex yang bisa diarsipkan atau dikirim ke *platform* menggunakan alat lain.
Show Sketch Folder	Membuka folder sketsa yang bersangkutan.
Include Library	Menambah pustaka ke sketsa dengan memasukkan *statemens* ***#include*** pada awal kode.
Add File	Menambah sebuah berkas sumber ke sketsa (disalin dari lokasinya). Berkas baru akan mucul dalam tab baru pada jendela sketsa. Berkas dapat dihapus dari sketsa menggunakan menu tab (ikon segitiga kecil dibawah ikon serial monitor pada sisikanan toolbar).

II.4.1.1 Sketchbook

Arduino Software (IDE) menggunakan konsep *sketchbook* yang mrupakan sebuah tempat standar untuk menyimpan program (sketsa). Sketsa-sketsa pada *sketchbook* dapat dibuka melalui opsi menu *File* > *Sketchbook* dari tombol Open pada *toolbar*.

Saat pertama kali menjalankan perangkat lunak Arduino, secara otomatis akan memuat sebuah direktori untuk *sketchbook*. Lokasi *sketchbook* dapat dilihat atau diganti menggunakan log *Preferences*.

11.4.1.2 Tab, Multiple Files and Compilation

Memungkinkan pengguna mengatur sketsa-sketsa dengan lebih dari satu berkas (tiap-tiapnya akan muncul pada tab baru). Hal ini dapat berupa berkas kode Arduino normal (tanpa ekstensi terlihat), berkas C (ekstensi .c), berkas C++ (ekstensi .cpp), atau berkas header (.h).

11.4.1.3 Pengunggahan

Sebelum mengunggah berkas, perlu dipilih item yang sesuai dari menu *Tools* > *Board* dan *Tools* > *Port*. Pada Mac, port serial kemungkinan berupa sesuatu seperti : ***/dev/tty.usbmodem241*** (untuk Uno atau Mega2560 atau Leonardo) atau ***/dev/tty.usbserial-1B1*** (untuk Duemilanove atau *board* USB sebelumnya) atau ***/dev/tty.USA19QW1b1P1.1*** (untuk *board* serial yang terhubung dengan adapter USB-to-serial Keyspan).

Pada Windows, kemungkinan COM1 atau COM2 (untuk *board* serial) atau COM4, COM5, COM7, atau yang lebih tinggi (untuk *board* USB), untuk menemukannya, lihat perangkat USB serial pada seksi port dari Windows Device Manager. Pada GNU/Linux bisa berupa ***/dev/ttyACMx, /dev/ttyUSBx*** atau mirip dengan itu. Setelah selesai dipilih *port* serial dan *board* yang benar, maka klik ikon *upload* pada *toolbar* atau melalui *Sketch* > *Upload*. *Board* Arduino yang sedang difungsikan akan otomatis ter-*reset* dan memulai pengunggahan. Untuk *board* yang lama (pra-Diecimilia), perlu menekan tombol *reset* pada *board* sebelum mulai mengunggah. Pada kebanyakan *board*, akan terlihat LED TX dan RX berkedip sepanjang sketsa diunggah. Arduino Software (IDE) akan menampilkan pesan ketika pengunggahan selesai, atau menampilkan *error*.

II.4.1.4 Pustaka (*Library*)

Pustaka menyediakan fungsionalitas ekstra untuk digunakan dalam sketsa, misalnya bekerja dengan perangkat keras atau memanipulasi data. Untuk menggunakan suatu pustaka dalam sketsa, pilih sub menu *Sketch* > *Import Library*. Hal ini akan memasukkan satu atau lebih *statements #include* pada awal sketsa dan mengkompilasi pustaka tersebut dengan sketsa pengguna. Karena pustaka-pustaka yang diunggah ke *board*, hal ini

menambah jumlah memori yang digunakan. Jika sketsa tidak lagi membutuhkan pustaka, hapus *statements #include* dari awal kode.

II.4.1.5 Perangkat Keras Pihak Ketiga

Dukungan untuk perangkat keras pihak ketiga dapat ditambahkan ke direktori *hardware* dari direktori *sketchbook*. *Platform* yang ter-*install* di sana bisa mengandung definisi *board* (yang muncul dalam menu *Board*), pustaka-pustaka inti, *bootloader*, dan definisi pemrogram. Untuk meng-*install* buat direktori *hardware*, kemudian *unzip* platform pihak-ketiga ke sub-direktorinya sendiri. Untuk melakukan pencopotan, hapus bagian direktori. Untuk detail pembuatan paket untuk perangkat keras pihak-ketiga, lihat spesifikasi perangkat keras pihak ketiga Arduino IDE 1.5.

II.4.1.6 Serial Monitor

Menampilkan data serial yang sedang dikirim dari *board* Arduino atau Genuino (*board* USB atau serial). Untuk mengirim data ke *board*, masukkan teks dan klik pada tombol ***send*** atau tekan *enter*. Pilih *baud rate* yang cocok dengan rate yang terlewat ke *Serial.begin* pada sketsa.

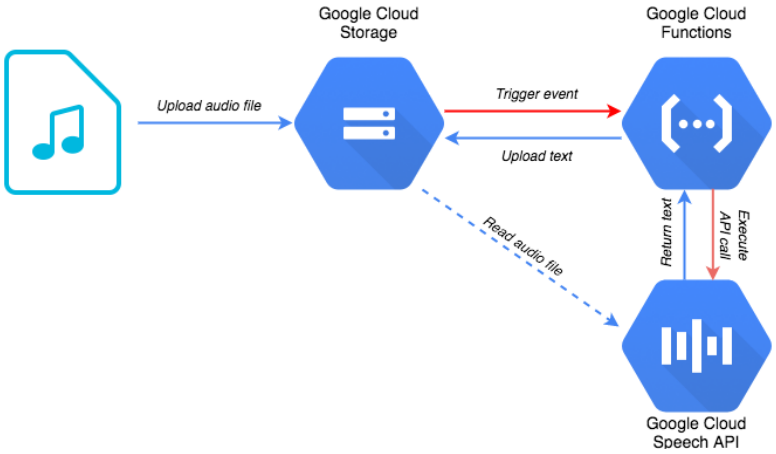
II.4.1.7 Pengaturan

Beberapa pengaturan dapat diatur dalam dialog *Preferences*. Pengaturan-pengaturan lain dapat ditemukan pada berkas *preferences*, yang lokasinya ditampilkan pada dialog *preferences*.

II.4.2 Google Cloud Speech API

II.4.2.1 *Cloud Speech-to-Text API*

Permintaan pengenalan sinkron *Speech-to-Text API* adalah metode paling sederhana untuk melakukan pengenalan pada data audio suara. Ucapan-ke-Teks dapat memproses hingga 1 menit data audio ucapan yang dikirim dalam permintaan sinkron. Setelah Ucapan-ke-Teks memproses dan mengenali semua audio, ia mengembalikan respons. Ucapan-ke-Teks biasanya memproses audio lebih cepat daripada waktu nyata, memproses 30 detik audio dalam rata-rata 15 detik. Dalam kasus kualitas audio yang buruk, permintaan pengenalan bisa memakan waktu lebih lama.



Gambar 2.3 Flow proses Speech-to-Text

Speech-to-Text memiliki tiga metode utama untuk melakukan pengenalan suara. Di antaranya adalah :

a. *Synchronous Recognition* (REST dan gRPC), yang mengirimkan data audio ke *Speech-to-Text API*, melakukan pengenalan pada data tersebut, dan mengembalikan hasil setelah semua audio diproses. Permintaan pengenalan sinkron terbatas pada data audio berdurasi 1 menit atau kurang.
b. *Asynchronous Recognition* (REST dan gRPC), yang mengirimkan data audio ke *Speech-to-Text API* dan memulai *Long Running Operation*. Dengan menggunakan operasi ini, pengembang dapat melakukan *polling* secara berkala untuk hasil pengenalan. Gunakan permintaan asinkron untuk data audio dengan durasi hingga 180 menit.
c. Pengenalan *Streaming* (khusus gRPC) melakukan pengenalan pada data audio yang disediakan dalam aliran dua arah gRPC. Permintaan *streaming* dirancang untuk tujuan pengenalan waktu nyata, seperti menangkap audio langsung dari mikrofon. Pengenalan *streaming* memberikan hasil sementara saat audio sedang ditangkap, memungkinkan hasil muncul, misalnya, saat pengguna masih berbicara. Permintaan berisi parameter konfigurasi serta data audio.

Permintaan *Speech-to-Text API* yang sinkron terdiri dari konfigurasi pengenalan suara, dan data audio. Sementara permintaan *Speech-to-Text API* asinkron, menggunakan metode *Long Running Recognize* dan akan memulai *Long Running Operation* (dari Operasi tipe).

II.4.2.2 *Cloud Text-to-Speech API*

Cloud Text-to-Speech API memungkinkan pengembang untuk membuat suara manusia sintetis yang terdengar alami sebagai audio yang dapat diputar. Pengguna dapat menggunakan file data audio yang dibuat

menggunakan *Cloud Text-to-Speech API* untuk memberi daya pada aplikasi atau menambah media seperti video atau rekaman audio (sesuai dengan Ketentuan Layanan *Google Cloud Platform* termasuk kepatuhan terhadap semua hukum yang berlaku).

Text-to-Speech API berfungsi untuk mengubah teks atau input *Speech Synthesis Markup Language* (SSML) menjadi data audio seperti MP3 atau LINEAR16 (pengodean yang digunakan dalam file WAV). *Text-to-Speech API* sangat ideal untuk aplikasi apa pun yang memutar audio ucapan manusia ke pengguna. Ini memungkinkan pengembang untuk mengubah string, kata, dan kalimat yang berubah-ubah menjadi suara seseorang yang berbicara hal yang sama.

Gambar 2.4 Flow proses Text-to-Speech

Proses menerjemahkan input teks ke dalam data audio disebut sintesis dan output sintesis disebut ucapan sintetis. *Text-to-Speech API* mengambil dua jenis input, yakni teks mentah atau data berformat SSML.

Proses sintesis ucapan menghasilkan data audio mentah sebagai string yang dikodekan base64. Pengembang harus mendekodekan string yang disandikan base64 ke dalam file audio sebelum aplikasi dapat memainkannya. Sebagian besar *platform* dan sistem operasi memiliki alat untuk mendekode teks base64 menjadi file media yang dapat diputar.

a. Suara

Text-to-Speech API menciptakan data audio mentah dari ucapan manusia yang alami. Artinya, ia menciptakan audio yang terdengar seperti orang yang berbicara. *Text-to-Speech API* memiliki beragam pilihan suara khusus yang tersedia untuk digunakan. Suara-suara berbeda didasarkan pada bahasa, jenis kelamin, dan aksen (untuk beberapa bahasa).

b. Suara WaveNet

Bersamaan dengan suara sintetis tradisional lainnya, *Text-to-Speech API* juga menyediakan suara premium yang dihasilkan WaveNet. Pengguna menemukan suara yang dihasilkan Wavenet lebih hangat dan mirip manusia daripada suara sintetis lainnya.

Perbedaan utama untuk suara WaveNet adalah model WaveNet yang digunakan untuk menghasilkan suara. Model WaveNet telah dilatih menggunakan sampel audio mentah dari manusia yang sebenarnya berbicara. Hasilnya, model-model ini menghasilkan ucapan sintetis dengan lebih banyak penekanan dan infleksi seperti manusia pada suku kata, fonem, dan kata-kata.

c. Pengaturan output audio lainnya

Selain suara, pengembang juga dapat mengonfigurasi aspek lain dari output data audio yang dibuat oleh sintesis ucapan. *Text-to-Speech API* mendukung konfigurasi kecepatan bicara, nada, volume, dan tingkat sampel hertz.

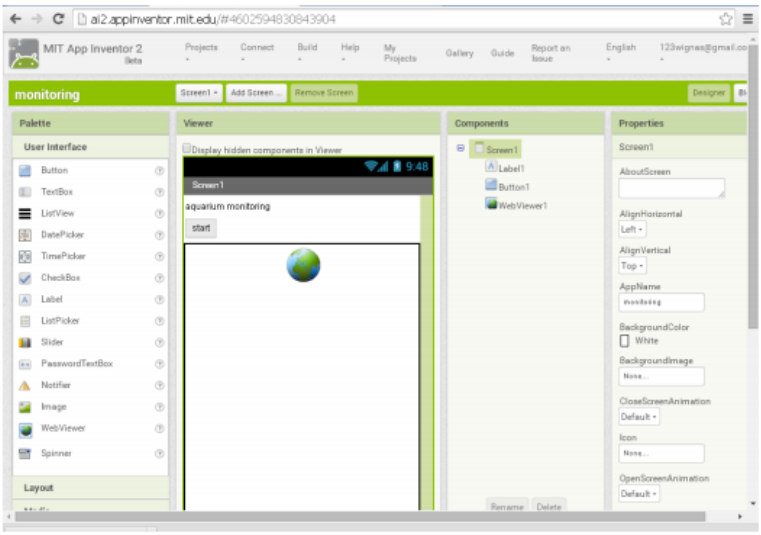
d. Dukungan *Speech Synthesis Markup Language* (SSML)

Pengembang dapat meningkatkan pidato sintetis yang dihasilkan oleh *Text-to-Speech API* dengan menandai teks menggunakan *Speech Synthesis Markup Language* (SSML). SSML memungkinkan pengguna untuk memasukkan jeda, pengucapan akronim, atau detail tambahan lainnya ke dalam data audio yang dibuat oleh *Text-to-Speech API*. *Text-to-Speech API* mendukung subset elemen SSML yang tersedia.

II.4.3 App Inventor 2 (AI2)

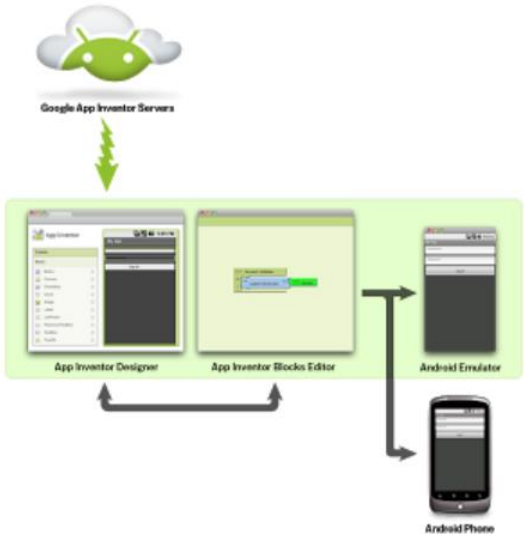
Merupakan sebuah *tool* untuk membuat aplikasi android. App Inventor kini dikembangkan oleh MIT, universitas yang bergerak di bidang teknologi. App Inventor awal mulanya dikembangkan oleh Google, namun diambil alih oleh MIT yang memegang kendali terhadap pengembangan *tools* App Inventor.

MIT App Inventor berbasis *visual block programming*, sehingga dapat membuat aplikasi tanpa kode satupun. *Visual block programming* merupakan pemrograman dengan menggunakan, menyusun, dan melakukan proses *drag-drops* terhadap blok-blok, yang menjadi simbol-simbol perintah dan fungsi *event handler* tertentu dalam membuat aplikasi. App Inventor tidak hanya untuk membuat suatu aplikasi, namun dapat juga digunakan untuk mengasah logika, seperti halnya menyusun sebuah puzzle. App inventor dibangun untuk pemula yang mulai belajar membuat aplikasi android. Tampilan utama pada MIT App Inventor dapat dilihat pada **Gambar 2.5.**



Gambar 2.5. Tampilan Utama Pada MIT APP Inventor2

Cara kerja dari MIT App Inventor adalah menggunakan *Framework Visual Programming* yang terkait dengan bahasa pemrograman Scratch dari MIT. Bahasa pemrograman tersebut secara spesifik merupakan implementasi dari *Open Block* yang didistribusikan oleh MIT Scheller Teacher Education Program yang diambil dari riset yang dilakukan oleh Ricarose Roque. App Inventor menggunakan *Kawa Language Framework* dan *Kawa's dialect* yang dicetuskan oleh Per Bothner dan didistribusikan sebagai bagian dari *GNU Operating System* oleh Free Software Foundation dan berfungsi sebagai *Compiler* yang menerjemahkan *visual block programming* untuk diimplementasikan pada platform Android. Diagram blok pada **Gambar 2.6** menggambarkam proses dari pembuatan aplikasi android.



Gambar 2.6 Blok Diagram Kerja Pada MIT App Inventor