DESIGN IMMERSIVE TOOL MULTISENSOR PADA *GAME* SEPEDA BERBASIS STATISTIK DAN KOMUNIKASI DATA *ASINKRON*

SKRIPSI

Oleh:

POGAL INDRA MUSSUGA

NIM. 08650106

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM MALANG
2014

DESIGN IMMERSIVE TOOL MULTISENSOR PADA *GAME* SEPEDA BERBASIS STATISTIK DAN KOMUNIKASI DATA *ASINKRON*

SKRIPSI

Diajukan Kepada:

Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:
POGAL INDRA MUSSUGA
NIM. 08650106

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM MALANG
2014

HALAMAN PERSETUJUAN

DESIGN IMMERSIVE TOOL MULTISENSOR PADA GAME SEPEDA BERBASIS STATISTIK DAN KOMUNIKASI DATA ASINKRON

SKRIPSI

Oleh:

POGAL INDRA MUSSUGA

NIM. 08650106

Telah Diperiksa dan Disetujui untuk Diuji:

Tanggal: 14 April 2014

Pembimbing I,	**Pembimbing II,**
Yunifa Miftachul Arif, M.T	**Fresy Nugroho, M.T**
NIP. 198306162011011004	**NIP. 197107222001011001**

Mengetahui,

Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian

NIP. 197404242009011008

HALAMAN PENGESAHAN

DESIGN IMMERSIVE TOOL MULTISENSOR PADA GAME SEPEDA BERBASIS STATISTIK DAN KOMUNIKASI DATA ASINKRON

SKRIPSI

Oleh:

POGAL INDRA MUSSUGA

NIM. 08650106

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S. Kom)
Tanggal 14 April 2014

Susunan Dewan Penguji:		**Tanda Tangan**
1. Penguji Utama	**: Fachrul Kurniawan, M. MT** **NIP. 19771020 200901 1 001**	**( )**
2. Ketua Penguji	**: Dr. M.Faisal, M.T** **NIP. 19740510 200501 1 007**	**( )**
3. Sekretaris	**: Yunifa Miftachul Arif, MT** **NIP. 19830616 201101 1 004**	**( )**
4. Anggota Penguji	**: Fresy Nugroho, M.T** **NIP. 19710722 200101 1 001**	**( )**

Mengetahui,

Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian

NIP. 197404242009011008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan dibawah ini:

Nama : Pogal Indra Mussuga

NIM : 08650106

Jurusan : Teknik Informatika

Fakultas : Sains dan Teknologi

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 14 April 2014

Yang membuat pernyataan,

Pogal Indra Mussuga

NIM. 08650106

MOTTO

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُوٓاْ إِذَا قِيلَ لَكُمۡ تَفَسَّحُواْ فِي ٱلۡمَجَٰلِسِ فَٱفۡسَحُواْ
يَفۡسَحِ ٱللَّهُ لَكُمۡۖ وَإِذَا قِيلَ ٱنشُزُواْ فَٱنشُزُواْ يَرۡفَعِ ٱللَّهُ ٱلَّذِينَ ءَامَنُواْ
مِنكُمۡ وَٱلَّذِينَ أُوتُواْ ٱلۡعِلۡمَ دَرَجَٰتٖۚ وَٱللَّهُ بِمَا تَعۡمَلُونَ خَبِيرٞ ١١

Hai orang-orang beriman apabila dikatakan kepadamu: "Berlapang-lapanglah dalam majlis", maka lapangkanlah niscaya Allah akan memberi kelapangan untukmu. Dan apabila dikatakan: "Berdirilah kamu", maka berdirilah, niscaya Allah akan meninggikan orang-orang yang beriman di antaramu dan orang-orang yang diberi ilmu pengetahuan beberapa derajat. Dan Allah Maha Mengetahui apa yang kamu kerjakan. (QS. Mujaadilah 58:11)

THE POWER OF DREAMS

BERMIMPILAH DAN WUJUDKAN IMAJINASIMU

PERSEMBAHAN

Karya tulis ini saya persembahkan kepada:

Ayahku Agus Amin Subagia dan Ibuku Mussripah

yang telah sukses mendidik kami

dengan penuh kasih sayang dan cinta,

dengan harapan di hati

dengan do'a yang terucap,

dengan kesungguhan usaha.

Adikku Galang Dinar Mussuga

yang telah memberikan dukungan atas semua ide - ide

dan gagasan yang telah diberikan

Tim game gowes,

Oxalie Triadmaja, Haris Budi Erwanto, M Agung Tarecha

yang telah solid bekerja sama dalam pembuatan Tugas

Akir ini. Dan tak lupa terimakasih untuk teman satu

angkatan 2008.

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh

Syukur Alhamdulillah penulis haturkan kehadirat Allah SWT yang telah melimpahkan Rahmat dan Hidayah-Nya, sehingga penulis dapat menyelesaikan studi di Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang sekaligus menyelesaikan skripsi ini dengan baik.

Sholawat serta salam semoga selalu terlimpahkan kepada Nabi besar kita Nabi Muhammad SAW yang telah membimbing kita dari zaman jahiliyah menuju zaman yang terang benderang yaitu Islam.

Selanjutnya penulis hanturkan ucapan terima kasih seiring do'a dan harapan *jazakumullah ahsanal jaza'* kepada semua pihak yang telah membantu terselesaikannya skripsi ini. Penulis sampaikan terima kasih kepada Ayah dan Ibu tercinta yang senantiasa memberikan do'a dan restunya kepada penulis selama ini, dan semua anggota tim *Game* Gowes yang selalu aktif dalam membantu dan berdiskusi sehingga skripsi ini dapat diselesaikan dengan baik dan atas semua kerja keras selama mengerjakan *game* ini, dan juga kepada segenap sivitas akademika UIN Maulana Malik Ibrahim Malang terutama Prof. Dr. H. Mudjia Rahardjo, M.Si, selaku Rektor UIN Maulana Malik Ibrahim Malang saat ini dan juga tak lupa Prof. Dr. Imam Suprayogo selaku mantan rektor periode

sebelumnya, Yunifa Miftachul A, MT dan Fresy Nugroho, M.T selaku dosen pembimbing skripsi, yang telah banyak memberikan pengarahan. Tak lupa kepada semua pihak yang turut mendukung sehingga skripsi ini dapat diselesaikan dengan baik.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih terdapat kekurangan dan penulis berharap semoga skripsi ini bisa memberikan manfaat kepada para pembaca khususnya bagi penulis secara pribadi.

Amin Ya Rabbal Alamin.

Wassalamu'alaikum Warahmatullahi Wabarakatuh

Malang, 14 April 2014

Penulis,

Pogal Indra Mussuga

DAFTAR ISI

KATA PENGANTAR viii
DAFTAR GAMBAR xi
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 3
1.4 Tujuan Penelitian 3
1.5 Manfaat Penelitian 4
1.6 Metode Penelitian 4
1.7 Sistematika Penulisan 6
BAB II TINJAUAN PUSTAKA 8
2.1 *Immersive Tool* 8
2.2 Statistik data Multisensor 9
2.3 Komunikasi data serial yang menggunakan USB A/B Asinkron 14
2.4 Mikrokontroler Arduino Duemilanove 16
2.5 ADC 18
2.6 Sensor 19
BAB III ANALISIS DAN PERANCANGAN 23
3.1 Kebutuhan Sistem 23
3.1.1 Kebutuhan *Platform* 23
3.1.2 Deskripsi Sistem 24
3.1.3 Kalibrasi Sensor 49
3.2 Perancangan Alur Proses 56
BAB IV HASIL DAN PERANCANGAN 60
4.1 Hasil Uji Coba 60
4.1.1 Hasil Kalibrasi 60
4.1.2 Hasil uji komunikasi serial Asinkron 65
4.1.3 Hasil uji coba statistik data 66
4.2 Koneksi Data Multisensor ke *Game* 69
4.4 Integrasi *Game* sepeda menggunakan *Immersive Tool* dengan Islam 74
BAB V PENUTUP 75
5.1 Kesimpulan 75
5.2 Saran 76
DAFTAR PUSTAKA 77

DAFTAR GAMBAR

Gambar 2. 1 *Immersiv Tool*....8
Gambar 2. 2 uji coba *Immersive Tool* 9
Gambar 2. 3 *potensiometer* dengan belokan kanan dan kiri 10
Gambar 2. 4 Ilustrasi 11
Gambar 2. 5 tanda sensor pada roda....12
Gambar 2. 6 *speedometer* untuk kecepatan 14
Gambar 2. 7 *Arduino Duemilanove*....16
Gambar 2. 8 berikut ini adalah IDE *Arduino*....17
Gambar 2. 9 dari infra merah (TX) dan *photo dioda* (RX)....19
Gambar 2. 10 aplikasi sensor garis dengan infra merah dan *photo dioda*....20
Gambar 2. 11 sensor potensio 21
Gambar 2. 12 tombol *push on.* 22
Gambar 3. 1 dari komunikasi *Arduino* dengan *game* menggunakan USB....25
Gambar 3. 2 kabel USB A/B....26
Gambar 3. 3 dari kaki soket USB....27
Gambar 3. 4 gambar proses pengiriman data serial *asinkron* 30
Gambar 3. 5 gambar proses penerima data serial *asinkron* 31
Gambar 3. 6 dasar paket UART: 1 Mulai bit, 8 bit data, 1 bit paritas dan 1 bit berhenti ..32
Gambar 3. 7 data titik pengambilan sampel oleh penerima *UART* 33
Gambar 3. 8 komfigurasi kaki-kaki *Arduino*....34
Gambar 3. 9 *Integrated Development Enviroment* 37
Gambar 3. 10 bentuk LED Infra merah....39
Gambar 3. 11 kurva tanggapan *Frekuensi* Sensor *Photo dioda*....40
Gambar 3. 12 proses sensor saat mendeteksi warna hitam dan putih 41
Gambar 3. 13 penempatan sensor kecepatan di sepeda 42
Gambar 3. 14 dari penempatan potensio unutk sensor kemudi dan pengereman....43
Gambar 3. 15 *potensiometer* dengan belokan kanan dan kiri 44
Gambar 3. 16 tombol *push on* 45
Gambar 3. 17 tanda sensor pada roda....46
Gambar 3. 18 rancangan alur proses sensor data....56
Gambar 3. 19 proses pengolahan data *multisensor*....57
Gambar 3. 20 Diagram Blok 58
Gambar 4. 1 hasil kalibrasi belok kiri 61
Gambar 4. 2 hasil kalibrasi belok kanan....61
Gambar 4. 3 kalibrasi sensor rem dalam keadaan tidak mengerem/ *low* 62
Gambar 4. 4 kalibrasi sensor rem dalam keadaan menekan rem/ *high* 62
Gambar 4. 5 kalibrasi pada titik putih 63
Gambar 4. 6 kalibrasi pada titik hitam 63
Gambar 4. 7 hasil uji coba tombol *push on* 64
Gambar 4. 8 hasil data *speedometer* dan sensor pada uji coba 66
Gambar 4. 9 hasil data *speedometer* dan sensor pada uji coba 67
Gambar 4. 10 hasil data *speedometer* dan sensor pada uji coba 67
Gambar 4. 11 hasil data *speedometer* dan sensor pada uji coba 68

ABSTRAK

Mussuga, Pogal Indra. 2014. **Design *Immersive Tool* Multisensor pada *Game* Sepeda Berbasis Statistika dan Komunikasi Data Asinkron.** Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
Pembimbing: Yunifa Miftachul Arif, MT (1), Fresy Nugroho, M.T (2)

Kata kunci : Imssersive Tool, Multisensor

Ada banyak permainan yang sering dimainkan baik dari kalangan anak-anak dan tak sedikit dari kalangan orang dewasa. Dan ada banyak jenis permainan yang menarik, dalam hal ini simulasi *Game* Sepeda juga bisa menjadi sarana bermain dan juga sebagai saranan untuk berolah raga.

Pada era sekarang tak sedikit juga *Game* yang kurang mendidik anak-anak untuk mengembangkan bakat mereka, hal ini tak luput dari arahan orang tua. Maka dari itu penulis mencoba untuk menciptakan Simulasi *Game* Sepeda yang dimana dapat dimainkan oleh para anak-anak dan juga orang dewasa.

Pada *game* simulasi sepeda akan berupaya sebaik mungkin untuk menampilkan visual yang semirip mungkin dengan keadaan sesungguhnya. Mulai dari mengayuh, membelokkan kemudi dan juga mengerem jika lintasan akan berbelok maupun menemui rintangan, dengan adanya simulasi *Game* Sepeda ini diharapkan pemain bisa merasakan seperti bermain sepeda pada dunia nyata.

Berdasarkan latar belakang tersebut penulis ingin menerapkan *Immersive Tool* Multisensor pada *game* sepeda, yang pada pada komunikasi data akan menggunakan Asinkron. Jadi suatu sepeda nantinya akan di berikan multisensor pada bagian roda (kecepatan), kemudi dan rem. Tool yang nantinya nantinya disajikan dalam bentuk sepeda yang didepannya terdapat monitor yang berguna untuk tampilan rute/ jalan, hasil akirnya akan berupa *game* yang dapat memberikan banyak mamfaat bagi yang menjalankan *(player)*.

ABSTRACT

Mussuga, Pogal Indra. , 2014. **Design of Multisensor *Immersive Tool* in Bike *Game*s Based on Statistics and Asynchronous Data Communications**. Thesis. Department of Informatics, Faculty of Science and Technology of the State Islamic University of Maulana Malik Ibrahim Malang.
Supervisor: Yunifa Miftachul Arif, MT (1), Fresy Nugroho, M.T (2)

Keywords: Imssersive Tool, Multisensor

There are many *game*s that are often played either of the children and not a few of the adult population. And there are many interesting types of *game*s, in this simulation *game* Bicycles can also be a means to play as well as the proposition for exercise.

In the current era , not a few less *game*s teach children to develop their talents , it did not escape from parental directives . Thus the authors try to create a simulation *game* where the bike that can be played by children and adults alike .

On a bike simulation *game* will strive our best to visual displays that as closely as possible to the real situation . Start of pedaling , steering and brakes deflect when the track will turn and meet obstacles , with the bike is expected simulation *game player*s can feel like playing a bicycle in the real world .

Based on this background the author wants to apply *Immersive Tool* Multisensory on bike *game* , which will use the Asynchronous data communication . So the bike will be given multisensor on the wheel (speed) , steering and brakes . Tool that will later be presented in the form of the front of the bike there is a useful monitor for the display / path , the result will be a *game* akirnya can provide many mamfaat for that run (the *player*)

BAB I

PENDAHULUAN

1.1 Latar Belakang

Simulasi merupakan salah satu perkembangan teknologi akhir-akhir ini yang sangat pesat perkembangannya, terutama dalam bidang komputer. Kecanggihan teknologi saat ini dapat mensimulasikan perangkat-perangkat diluar komputer, dan disimulasikan kedalam komputer dalam bentuk virtual.

Salah satu perkembangan teknologi saat ini yang digemari oleh masyarakat luas yaitu permainan (*game*). *Game* merupakan sebuah alat untuk bermain dan *refreshing* yang sangat berkembang dalam lingkungan masyarakat. *Game* atau permainan di dalam komputer sebagian juga merupakan simulasi dari bentuk-bentuk nyata dalam kehidupan manusia. Banyak sekali *game* yang diciptakan semirip mungkin dengan kejadian nyata dan juga dari yang tersulit dan termudah.

Immersive Tool adalah sebuah alat kontrol yang dipergunakan untuk mengendalikan suatu objek didalam dunia virtual. Pemakaian *Immersive Tool* pada suatu *game* ditujukan untuk memfisualisasikan keadaan seperti sesungguhnya, hal ini dapat menjadikan suatu *game* lebih menantang dan menarik untuk dimainkan.

Game menjadi salah satu pilihan utama mengisi waktu senggang setelah beraktifitas. Citra *game* di masyarakat masih dipandang sebagai media yang menghibur dibanding sebagai media pembelajaran. Sifat dasar *game* yang menantang, membuat ketagihan dan menyenangkan bagi mereka yang menyukai permainan *modern* ini dapat berdampak *negatif* apabila yang dimainkan adalah *game* yang tidak bersifat edukatif.

Dari berbagai hal di atas, peneliti tertarik untuk membuat *game* seperti dengan keadaan sesunggunya dengan menggunakan *Immersive Tool*. Oleh karena itu diharapkan dengan adanya *game* sepeda yang menggunakan *Immersive Tool* ini pemain bisa mengira bermain *game* ini sama dengan bermain bersepeda pada umumnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan dapat dirumuskan rumusan masalahnya sebagai berikut:

1. Bagaimana model lingkup *Immersive Tool* untuk menyajikan perilaku dalam dunia nyata seperti belok, mengayuh dah mengerem?
2. Bagaimana *game* sepeda yang menggunakan *Immersive Tool* bisa menjadi sarana untuk berlatih naik sepeda?
3. Bagaimana komunikasi data dari *Immersive Tool* ke PC?

1.3 Batasan Masalah

Untuk menjaga fokus dari penelitian ini, maka beberapa batasan yang diberikan adalah sebagai berikut:

1. Sepeda yang dipakai adalah sepeda yang masih berfungsi baik dari segi pedal maupun kemudi.
2. *Immersive Tool* yang ada pada *game* sepeda ini hanya untuk 1 (satu) pemain saja.
3. *Game* sepeda yang menggunakan *Immersive Tool* diharapkan pemain bisa merasakan seperti bersepeda pada umumnya, baik dari segi mengngayuh dan berbelok begitu juga mengerem.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah membuat *game* sepeda yang dimana pemain bisa merasakan seperti menaiki sepeda asli. *Game* sepeda menggunakan *Immersive Tool* ini diharapkan bisa memberikan suasana yang berbeda dalam bermain *game* pada umumnya, dikarenakan *game* ini menggunakan sepeda asli untuk alat kendali. *Game* ini juga diupayakan sebagai sarana dan prasarana edukasi bagi anak-anak dan orang dewasa untuk berlatih sepeda dan untuk mengetes kelihaian pemain dalam mengendarai sepeda dalam dunia nyata.

1.5 Manfaat Penelitian

Beberapa mamfaat yang bisa diambil dari penelitian ini adalah:

1. Secara umum *game* ini bisa memberikan pelatihan dasar bagi pemain yang belum bisa atau sama sekali belum pernah bersepeda, dikarenakan takut jatuh saat masih kecil.
2. Sebagai sarana bermain dan berlatih dalam bersepeda.
3. *Game* ini juga bisa dijadikan prasarana untuk berolahraga.

1.6 Metode Penelitian

Tahapan yang dilakukan dalam proses penelitian dari desain ***Immersive Tool Multisensor* pada *game* sepeda berbasis statistik dan komunikasi data *asinkron*** adalah sebagai berikut:

1. **Studi Literatur**
    1. Pengumpulan informasi tentang *imersive tool*
    2. Pengumpulan *software* yang diperlukan.
    3. Pengumpulan informasi untuk spesifikasi *hardware* yang dibutuhkan.
2. **Perumusan Masalah**
    1. Menganalisa *hardware* dan *software* yang digunakan *user*.
    2. Menganalisa kebutuhan *user*.
    3. Menganalisa waktu pembuatan sistem.

3. **Analisis**

   1. Identifikasi dan Desain Sistem.
   2. Menganalisa *Immersive Tool*, *metode asinkron* dan bahasa pemrograman *C* pada mikrokontroler.
   3. Menganalisa *sistem Immersive Tool* pada *game* sepeda.

4. **Perancangan Sistem**

   1. Pembuatan desain fungsi yang digunakan.
   2. Pembuatan desain *input*.
   3. Pembuatan desain proses.
   4. Pembuatan desain *output*.
   5. Pembuatan desain sepeda dan penempatan sensor yang dibutuhkan.

5. **Pembuatan Hardware**

   1. Merancang disain *Immersive Tools.*
   2. Menyiapkan peralatan yang diperlukan seperti sepeda, mikrokontroler, sensor LED dan *potensiometer*, timah, dll.

6. **Pembuatan Software**

   1. Pembuatan *game* berupa dunia *virtual* sebagai medan.
   2. Penggabungan control *Immersive Tool* dengan *game* melalui mikrokontroler.
   3. *Finishing* visualisasi.
   4. Implementasi program ke Laptop/PC.

7. **Tahap Ujicoba dan Evaluasi**

   Uji coba dilakukan sampai sistem benar-benar *ready to use*, kekurangan yang terjadi diperbaiki dalam lingkup batasan masalah. Evaluasi dilakukan untuk mengetahui apakah sistem yang dibangun sudah sempurna apa masih perlu perbaikan lagi.

8. **Dokumentasi**

   Penulisan laporan skripsi merupakan dokumentasi dari keseluruhan pelaksanaan penelitian. Diharapkan dokumentasi penelitian berguna dan bermanfaat untuk penelitian dan pengembangan lebih lanjut.

1.7 Sistematika Penulisan

Penulisan skripsi ini tersusun dalam lima bab dengan sistematika penulisan sebagai berikut :

- BAB I Pendahuluan

  Bab ini berisi pembahasan tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

- BAB II Tinjauan Pustaka

  Pada Bab ini menjelaskan tentang teori-teori yang terkait dengan permasalahan yang akan diambil.

- BAB III Analisis dan Perancangan

  Bab ini menjelaskan tentang perancangan sepeda yang akan dibuat yang meliputi perancangan perangkat, yaitu penempatan sensor kecepatan, kemudi dan juga pengereman. Sedangkan IDE yang digunakan sampai pada pembuatan *game*.

- BAB IV Hasil dan Pembahasan

  Bab ini berisi pembahasan tentang pengiriman data dari sensor sampai diolah oleh mikrokontroler yang kemudian dikirim ke komputer menggunakan kominikasi dara serial Asinkron menggunakan USB. Serta akan dilakukan pengujian pada *game* yang dimana untuk mengetahui apakah sensor-sensor bisa berjalan dengan baik dan juga kalibrasi sensor, untuk memulai awal permaianan.

- BAB V Penutup

  Bab ini berisi tentang kesimpulan dan saran yang diharapkan dapat bermanfaat untuk pengembangan *game* selanjutnya.

- DaftarPustaka

  Berisikan seluruh materi referensi dalam penelitian skripsi ini, yang dimana tercantumkan dalam bab ini.

- Lampiran

  Berisikan lampiran data pendukung yang bertujuan untuk melengkapi uraian yang disajikan dalam bagian utama yang akan ditempatkan dibagian ini.

BAB II

TINJAUAN PUSTAKA

2.1 *Immersive Tool*

Immersive Tools adalah sebuah alat *control* yang dipergunakan untuk mengendalikan objek simulasi di dalam dunia *virtual*, misalnya sebuah *game Motoracer* di *Timezone* yang menggunakan model sepeda motor sebagai pengendali permainan. Dalam membuat *Immersive Tool* pada *game* sepeda ini juga akan memakai sepeda pada umumnya, hal ini ditujukan untuk membuat pemain untuk bisa merasa bermain *game* seperti kejadian bersepeda pada biasanya.

Gambar 2. 1 *Immersiv Tool*

Pada gambar di atas merupakan gambaran dari *Immersive Tool* pada *game* simulasi sepeda, dimana terdapat motor yang sudah dirangkai sensor dan mikrokontroler yang terhubung pada *game*.

Gambar 2. 2 uji coba *Immersive Tool*

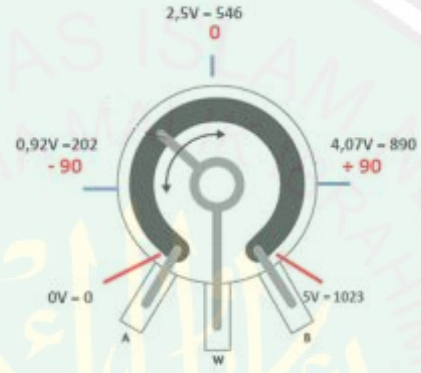
Pada gambar di atas pemakaian yang mengayuh untuk meningkatkan kecepatan dan juga kemudi untuk berbelok, selain itu pengereman jika mau berbelok dan jika ada halangan.

2.2 Statistik data Multisensor

Statistika adalah ilmu yang mempelajari bagaimana merencanakan, mengumpulkan, menganalisis, menginterpretasi, dan mempresentasikan data. Singkatnya, statistika adalah ilmu yang berkenaan dengan data. Statistika merupakan ilmu yang berkenaan dengan data, sedang statistik adalah data, informasi, atau hasil penerapan algoritma statistika pada suatu data.

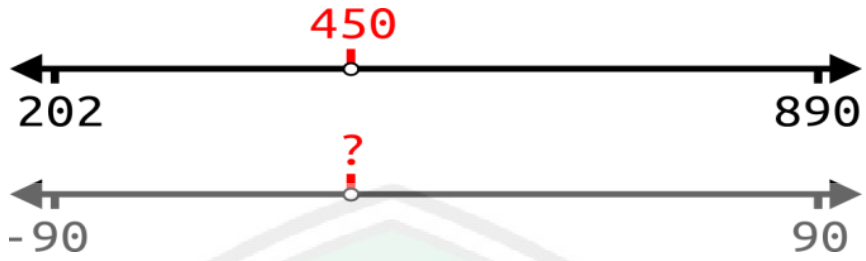
Dalam *game* ini kegunaan statistika untuk kalibrasi data dari sensor untuk ditampilkan pada monitor di *game*, hal ini ditujukan untuk menampilkan hasil

data *real* dari sensor. Maka dari itu untuk tampilan sensor kemudi diharapkan nantinya akan berupa *speedometer* sudut belokan dalam di *game*. Berikut ini gambaran sudut dan tegangan dari *potensiometer* yang digunakan untuk sensor kemudi.



Gambar 2. 3 *potensiometer* dengan belokan kanan dan kiri

Pada gambar ditunjukan bahwa pada posisi 0°(2,5V-546) maka posisi kemudi tepat lurus, sedangkan pada posisi -90°(0,92V-202) maka kemudi belok kiri dan 90°(4,7V-890) akan belok kanan. Maka dari itu untuk belok kiri nanti dari posisi 0 tegangan akan semakin rendah, begitu juga sebaliknya jika belok kanan maka tegangan akan tambah besar dari posisi 0 dan daya yang digunakan adalah 5V dan 10bit(0-1023) (Ashaulo, 2012). Misal pemain berbelok kekiri yang dimana diketahui bahwa nilai 10bit adalah 0-1023, dan disini pemain berbelok pada titik 450, maka dapat dilakukan dengan cara berikut.



Gambar 2. 4 Ilustrasi

Maka dapat dihitung.

$$titikOut = lowOut + (valueIn - lowIn) * \frac{highOut - lowOut}{highIn - lowIn}$$

dimana :

titikOut = titik hasil

lowOut = titik batas bawah hasil konversi yaitu -90

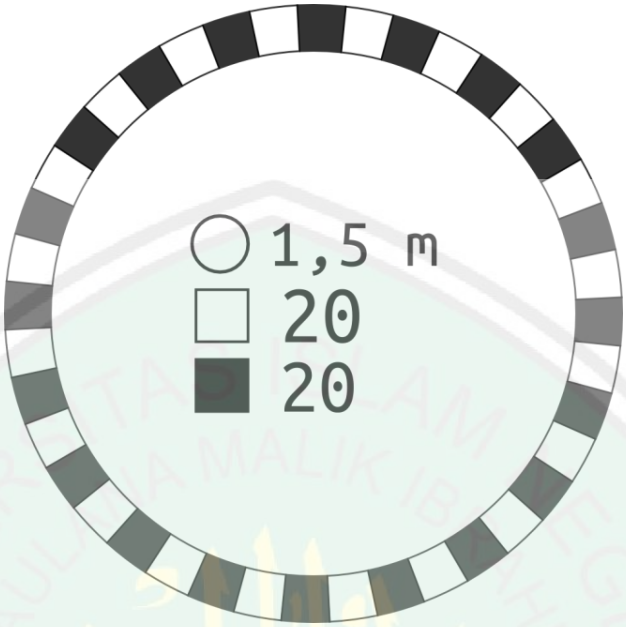
highOut = titik batas atas hasil konversi yaitu 90

lowIn = titik batas bawah titik asal dalam contoh ini adalah 202

highIn = titik batas atas titik asal dalam contoh ini adalah 890

valueIn = titik asal yang akan dikonversi

Sedangkan penghitungan kecapatan akan diperoleh dari berapa kali sensor membaca titik hitam (0) dan titik putih (1) dalam tiap detik. Maka dari itu akan diketahui kecepatan tiap detik, maka data yang yang ditampilkan di *speedometer* nanti merupakan data kecepatan *rotasi* roda, yang diperoleh dari:



Gambar 2. 5 tanda sensor pada roda

Pada gambar di atas dijelaskan bahwa tanda putih (1) berjumlah 20 dan keliling roda 1,5 meter, sedangkan untuk menghitung kecepatan terdapat pada program di mikrokontroler. Yaitu:

$$speed = \frac{count\ A}{jumlah\ tanda} \times 1.000 \times waktu\ sampling$$

Di mana:

Count A: jumlah baca tanda tiap detik

Jumlah tanda: jumlah data pada roda

1.000: waktu sampling menggunakan detik

Setelah data dalam satuan detik muncul maka nanti pada program mikrokontroler dimasukkan rumus:

$$AVG = \frac{kecepatan\ 1 + \cdots + kecepatan\ N}{N}.$$

Dimana kecepatan 1- kecepatan N merupakan hasil dari hitung kecepatan tiap detik, dan N merukapan jumlah dati data.

Dimana rps (*rotation per second*).

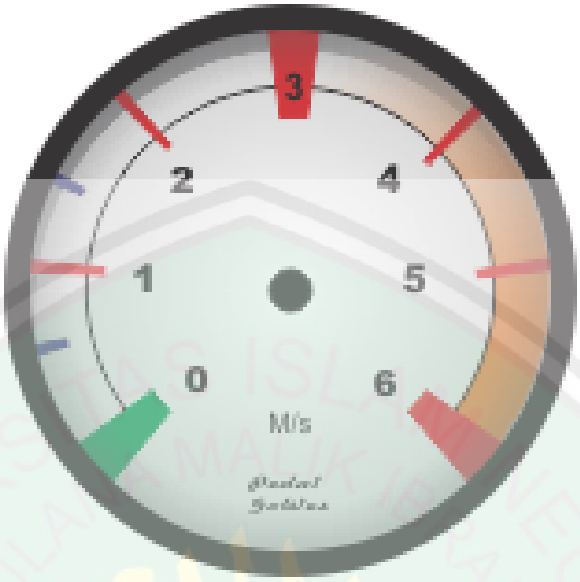
Dari data di atas dapat disimpulkan kecepatan minimal = 0, sedangkan untuk mencari kecepatan maksimal dapat diinisialisasi dengan.

```
max=0
        for(int i=1; i<=5; i++)
          {
              if(V(i)>max)
              {
                    Max= V(i)
              }
        }
```

Potongan kode di atas adalah rumus mencari kecepatan maksimal. Jika sudah didapat nilai per detik dari sensor maka akan mencari kecepatan jarak, menggunakan rumus

$$kecepatan = rps \times keliling\ roda$$

Maka hasil dari kecepatan jarak ini yang akan diperoleh berapa jarak yang telah dilalui *player* dalam bermain *game* menggunakan *Immersive Tool*. Dan hasil dari data kecepatan nantinya akan ditampilkan pada *speedometer* yang terdapat pada monitor *game*.



Gambar 2. 6 *speedometer* untuk kecepatan

Nantinya tampilan *speedometer* akan berada pada layar *game*, ini bertujuan untuk data statistik dari sensor kecepatan yang selalu berubah ketika *player* memainkan *game*. Dari *speedometer* ini nantinya bisa dilihat kecepatan rata-rata dan kecepatan maksimal.

2.3 Komunikasi data serial yang menggunakan USB A/B Asinkron

Komunikasi serial adalah komunikasi yang mengantarkan data digital secara bit per bit secara bergantian melalui media *interface* serial. Dalam *game* sepeda yang menggunakan *Immersive Tool* ini menggunakan pengiriman data serial menggunakan USB A/B. Didalam komunikasi serial terdapat 2 macam cara komunikasi data serial yaitu ***Sinkron*** dan ***Asinkron***. Pada komunikasi ***data serial Sinkron***, data dikirim dalam bentuk berkelompok (blok) dalam kecepatan yang teteptanpa bit awal dan bit ahir. Awalan blok *(start block)* dan akhiran *blok (stop block)* diidentifikasikan dalam bentuk *bytes* dengan susunan yang spesifik. *Clock*

pada penerima dioprasikan secara terus menerus dan dikunci agar sama dengan *clock* yang diterima pengirim. Sedangkan pada komunikasi ***data serial Asinkron*** tidak memiliki bit awalan dan akhiran, maka transmisi tak *sinkron* memiliki kedua bit tersebut. Pada transmisi ini, informasi akan diuraikan menjadi karakter dan masing masing karakter tersebut memiliki bit yang diidentifikasikan sebagai awalan blok *(start block)* dan bit akhiran blok *(stop block).*

Pada *game* sepeda yang menggunakan *Immersive Tool* ini memakai data serial ***Asinkron,*** Pengiriman data *Asinkron* ini lebih sederhana dibandingkan dengan pengiriman data sinkron karena hanya isyarat data saja yang dikirimkan. *Clock* penerima dibangkitkan secara lokal di dalam penerima dan tetap dijaga agar sesuai dengan *clock* pengirim. Bit awal dan bit akhir yang dikirimkan tidak membawa informasi tetapi hanya menunjukkan awal dan akhir setiap karakter.

Transmisi *Asinkron* digunakan apabila pengiriman data dilakukan 1 karakter setiap kali pengiriman. Transmisi ini dilakukan dengan cara memberikan bit awal *(start bit)* pada setiap awal pengiriman karakter dan di akiri dengan bit akir *(stop bit).*

Definisi metode *Asinkron* sebagai berikut:

. Pengiriman data dilakukan 1 karakter setiap kali, sehingga penerima harus melakukan sinkronisasi agar bit data yang dikirim dapat diterima dengan benar.

· Trasmisi keccepatan tinggi

· 1 karakter dengan yang lainnya tidak ada waktu antara yang tetap

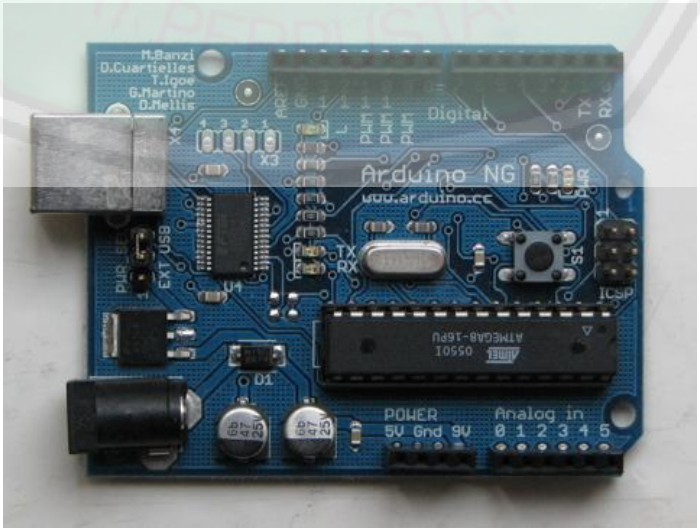
· Bila terjadi kesalahan maka 1 blok data akan hilang

- Membutuhkan *start pulse / start bit* (tanda mulai menerima bit data)
- *Idle transmitter* = '1' terus menerus, sebaliknya '0'.
- Tiap karakter diakhiri dengan *stop pulse / stop bit*.
- Dikenal sebagai start-*stop transmission*.

Pengiriman data *Asinkron* berupa *Clock* penerima dibangkitkan secara lokal di dalam penerima dan tetap dijaga agar sesuai dengan *clock* pengirim. Bit awal dan bit akir yang dikirim tidak membawa informasi tetapi hanya menunjukkan awal dan akir setiap karakter.

2.4 Mikrokontroler Arduino Duemilanove

Duemilanove adalah mikrokontroler CMOS 8-bit berarsitektur AVR RISC yang memiliki 8K Bytes In-System *Programmable Flash*. Mikrokontroler dengan konsumsi daya rendah ini mampu mengeksekusi instruksi dengan kecepatan maksimum 16 MIPS pada frekuensi 16MHz. Berikut adalah fitur selengkapnya dari AVR *Duemilanove*.



Gambar 2. 7 *Arduino Duemilanove*

Kelebihan dari *arduino Duemilanove:*

- Tidak perlu perangkat *chip programmer* karena di dalamnya sudah ada *bootloader* yang akan menangani *upload* program dari komputer.
- Sudah memiliki sarana komunikasi USB, sehingga pengguna Laptop yang tidak memiliki *port* serial/RS323 bisa menggunakan nya. Sambungan dari komputer ke *board Arduino* menggunakan USB, bukan serial atau *parallel port*. Sehingga akan mudah menghubungkan *Arduino* ke PC (*Personal Computer*) atau laptop yang tidak memeliki serial atau *parallel port*.
- Bahasa pemrograman relatif mudah karena *software Arduino* dilengkapi dengan kumpulan *library* yang cukup lengkap.
- Memiliki modul siap pakai (*shield*) yang bisa ditancapkan pada *board Arduino*. Misalnya *shield GPS, Ethernet, SD Card.*

1.4.1 Arduino Development Environment

Gambar 2. 8 berikut ini adalah IDE *Arduino*

Gambar di atas adalah lingkungan pengembangan *Arduino,* gambar di atas menjelaskan sebuah lingkungan pengembangan mencakup *toolbar*, serangkai menu, teks editor dan baris perintah konsol. Kode ditulis dalam *Arduino* disebut *sketsa,* dan memiliki ekstensi file *.ino. Arduino IDE* menggunakan alat baris perintah untuk mengkompilasi dan meng-*upload* kode sumber ke mikrokontroler. Selama proses kompilasi kode sumber, IDE memeriksa ukuran kode sumber terhadap memori yang tersedia dalam mikrokontroler. Jika kode sumber lebih besar dari memori yang tersedia, akan muncul bendera pesan kesalahan di konsol.

2.5 ADC

ADC adalah singkatan dari *Analog to Digital Converter*. Yang berfungsi untuk mengubah data analog menjadi data digital, sehingga dapat diproses oleh mikrokontroler. Hal-hal yang juga perlu diperhatikan dalam penggunaan ADC adalah tegangan maksimum yang dapat dikonversikan oleh ADC dari rangkaian pengkondisi sinyal, resolusi, pewaktu eksternal ADC, tipe keluaran, ketepatan dan waktu konversinya.

Beberapa karakteristik penting ADC:

1. Waktu konversi
2. Resolusi
3. Ketidak linieran
4. Akurasi

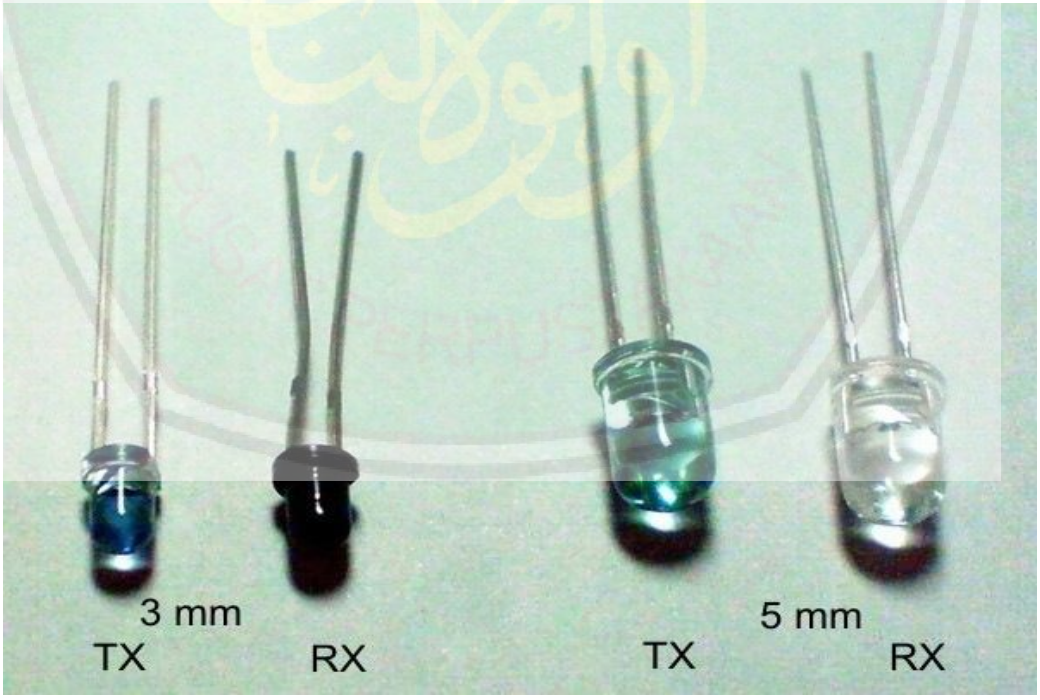
Ada banyak cara yang dapat digunakan untuk mengubah sinyal analog menjadi sinyal digital yang nilainya proporsional. Jenis ADC yang biasa

digunakan dalam perancangan adalah jenis *successive approximation convertion* atau pendekatan bertingkat yang memiliki waktu konversi jauh lebih singkat dan tidak tergantung pada nilai masukan analognya atau sinyal yang akan diubah.

2.6 Sensor

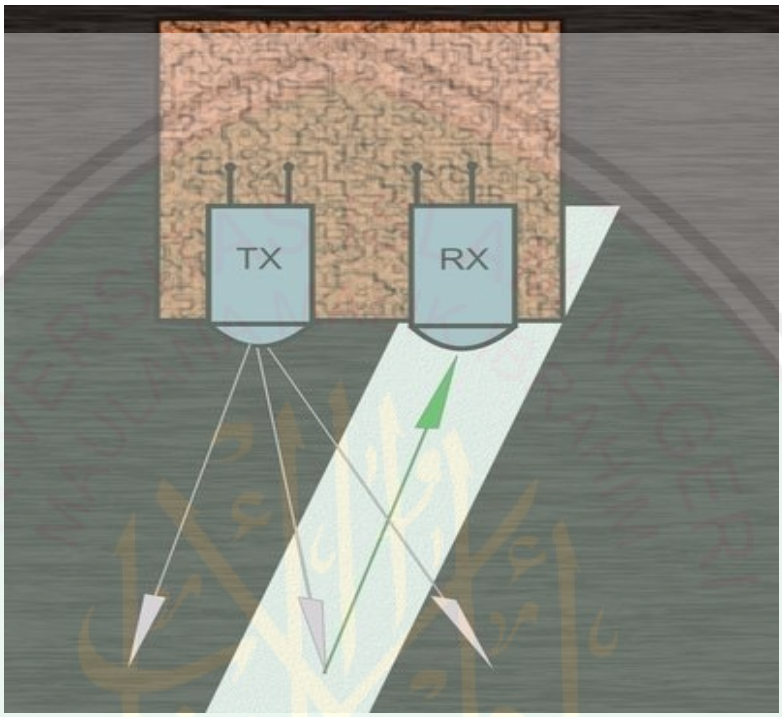
2.6.1 Sensor Kecepatan Menggunakan LED Infra Merah dan Photo dioda

Infra Red Adalah komponen elektronik yang dapat memancarkan sinar infra merah dengan jarak yang pendek dan tidak terlihat oleh mata. Contoh cahaya infra merah yaitu : cahaya api. Sedangkan *photo dioda* adalah komponen elektronika yang dapat menangkap sinar infra merah dan mengubahnya kedalam sinyal listrik berupa arus listrik (biasanya penggunaannya sebagai saklar).



Gambar 2. 9 dari infra merah (TX) dan *photo dioda* (RX)

Berikut adalah contoh gambar proses sensor kecepatan menggunakan infra merah dan *photo dioda* :

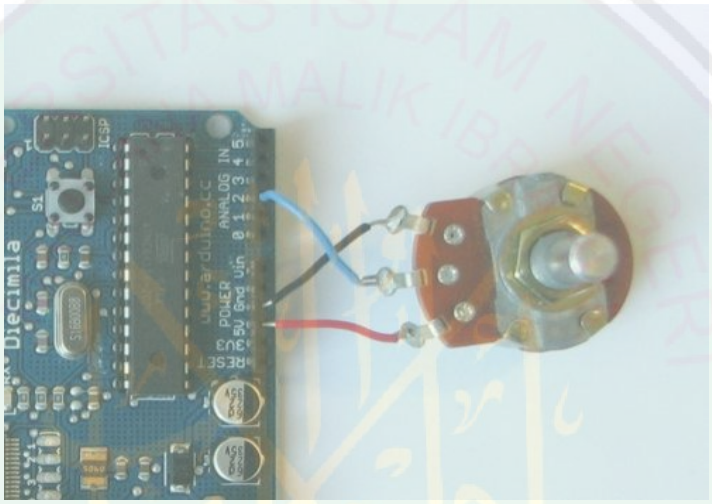


Gambar 2. 10 aplikasi sensor garis dengan infra merah dan *photo dioda*

Dari gambar di atas bisa dilihat bahwa infra merahakan memancarkan sinar dan akan ditangkap oleh *photo dioda*, dan penggunaan dalam sensor kecepatan maka nantinya ketika inframerah melewati warna putih maka sensor *photo dioda* akan menangkap dengan nilai 1 dan begitu juga sebaliknya ketika sensor infra merah melewati garis warna hitam maka sensor *photo dioda* tidak akan bisa mendeteksi (0). *Photo dioda* akan aktif apabila tidak terkena cahaya dari led infra merah. Antara led dan *photo dioda* dipisahkan oleh jarak. Jauh dekatnya jarak memengaruhi besar intensitas cahaya yang diterima oleh *photo dioda*. Pada bagian sensor ini terdapat sebuah LED infra merah (IR LED) yang berfungsi untuk mengirimkan sinyal kepada *receiver*. Pada *transmitter* dibangun dari sebuah

LED infra merah. Jika dibandingkan dengan menggunakan LED biasa, LED infra merah memiliki ketahanan yang lebih baik terhadap sinyal tampak. Cahaya yang dipancarkan oleh LED infra merah tidak terlihat oleh mata telanjang.

2.6.2 Sensor Kemudi dan Pengereman



Gambar 2. 11 sensor *potensiometer*

Sensor yang digunakan ialah *potensiometer*. *Potensiometer* adalah *resistor* tiga terminal dengan sambungan geser yang membentuk pembagi tegangan dapat disetel. Jika hanya dua terminal yang digunakan (salah satu terminal tetap dan terminal geser), *potensiometer* berperan sebagai *resistor variabel* atau *Rheostat*. *Potensiometer* biasanya digunakan untuk mengendalikan peranti elektronik seperti pengendali suara pada penguat. *Potensiometer* yang dioperasikan oleh suatu mekanisme dapat digunakan sebagai *transduser*, misalnya sebagai sensor *joystick*.

2.6.3 Tombol Push On

Tombol *push on* berguna ketika *player* menabrak pembatas dan ingin mundur ketika salah berbelok. Setiap *player* tentunya mempunyai trik ataupun cara tersendiri dalam memainkan suatu permainan, dan tombol ini disiapkan jika pemain ingin sepeda berjalan mundur.

Gambar 2. 12 tombol *push on.*

Berikut gambaran dari tombol *push on* dan dalam fungi di *game* ini untuk mundur, tombol *push on* dirasa cocok karena ketika *player* ingin bergerak mundur maka cukup menekan tombol dan sepeda pun akan mundur pada tampilan layar.

BAB III

ANALISIS DAN PERANCANGAN

3.1 Kebutuhan Sistem

Kebutuhan *Immersive Tool* yang diperlukan sebagai tahapan dasar analisis sistem yang akan dibangun, yaitu meliputi kebutuhan *platform*, diskripsi sistem, perancangan proses.

3.1.1 Kebutuhan *Platform*

Analisis kebutuhan merupakan analisis terhadap komponen-komponen yang digunakan untuk membuat sistem. Analisis kebutuhan ini terbagi menjadi dua macam, yaitu komponen perangkat lunak dan perangkat keras sebagai berikut.

1. Perangkat Lunak

   Perangkat lunak adalah istilah umum untuk data yang diformat dan disimpan secara digital, termasuk program komputer, dokumentasinya, dan berbagai informasi yang bisa dibaca dan ditulis oleh komputer. Perangkat lunak yang di gunakan dalam penelitian ini adalah:

   a. *Cadsoft Eagle*.

   b. *Arduino Development Environment*.

   c. *Google Sketchup v8.0.*

   d. *Windows 7 Ultimate Edition x64*.

2. Perangkat Keras

Perangkat keras yang dibutuhkan dalam penelitian ini adalah:

a. Sepeda.
b. Mikrokontroler *Arduino Duemilanove*.
c. Sensor (LED, *Potensiometer*, tombol *push on*)
d. PC/ laptop.

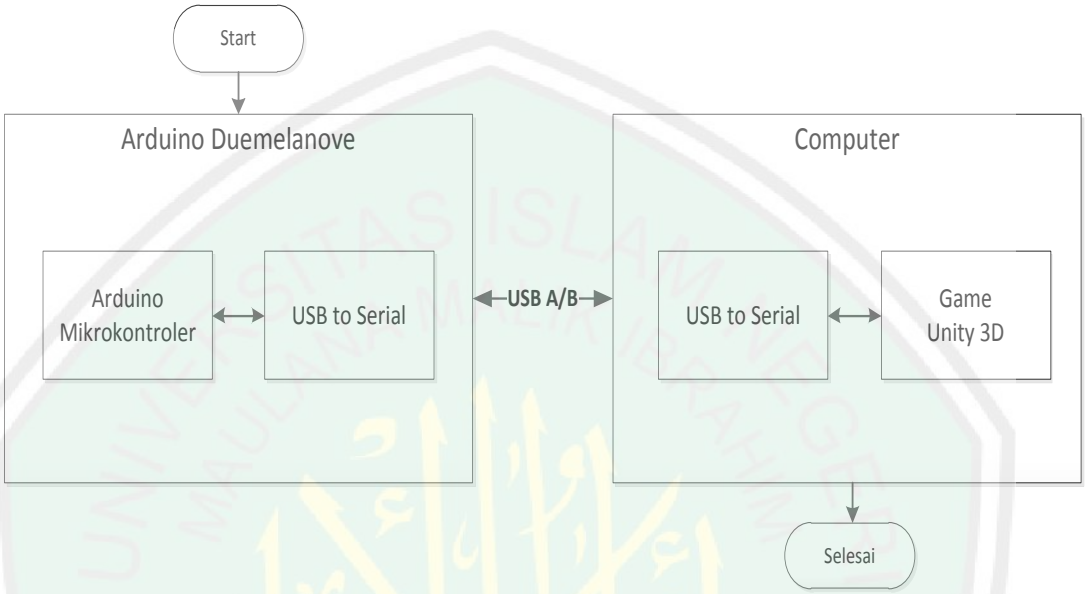
Untuk *spesifikasi hardware* dan *software* pada pc ataupun laptop, sudah mumpuni dan sudah di coba secara langsung untuk menjalankan *game* simulasi, hal yang berpengaruh dan menjadi prioritas adalah pada spesifikasi *hardware*, yang harus setara atau lebih pada spesifikasi di atas, karena hal ini berpengaruh pada pengolahan grafis yang menggunakan animasi 3D.

3.1.2 Deskripsi Sistem

3.1.2.1 Komunikasi data serial menggunakan USB A/B *Asinkron*

Komunikasi serial adalah komunikasi yang pengiriman datanya per-*bit* secara berurutan dan bergantian. Komunikasi ini mempunyai suatu kelebihan yaitu hanya membutuhkan satu jalur dan kabel yang sedikit dibandingkan dengan komunikasi paralel. Pada prinsipnya komunikasi serial merupakan komunikasi dimana pengiriman data dilakukan per bit sehingga lebih lambat dibandingkan komunikasi paralel, atau dengan kata lain komunikasi serial merupakan salah satu metode komunikasi data di mana hanya satu bit data yang dikirimkan melalui seuntai kabel pada suatu waktu tertentu. Pada dasarnya komunikasi serial adalah kasus khusus komunikasi paralel dengan nilai n = 1, atau dengan kata lain adalah

suatu bentuk komunikasi paralel dengan jumlah kabel hanya satu dan hanya mengirimkan satu bit data secara simultan.



Gambar 3. 1 dari komunikasi *Arduino* dengan *game* menggunakan USB

Dalam pengiriman data menggunakan pengiriman data serial dengan USB ***(****Universal Serial Bus****)***, kabel USB adalah sebuah kabel serabut yang digunakan untuk menyambung perangkat ke dalam CPU atau komputer. Kabel USB memiliki dua ujung yang ujung satu berupa soket USB yang nantinya akan dicolokkan ke *port* USB pada komputer dan yang di ujung lain meiliki adapter yang akan dicolokkan pada mikrokontroler. Fungsi kabel USB yaitu sesuai namanya USB yang merupakan singkatan dari *Universal Serial Bus*, gunanya untuk mentransmisikan data dari mikrokontroler ke dalam CPU dan kemudian data informasi tersebut akan diproses lebih lanjut oleh komputer.

Gambar 3. 2 kabel USB A/B

Gambar di atas merupakan model kabel USB, Desain USB ditujukan untuk menghilangkan perlunya penambahan *expansion card* ke ISA komputer atau bus PCI, dan memperbaiki kemampuan *plug-and-play* (pasang-dan-mainkan) dengan memperbolehkan peralatan-peralatan ditukar atau ditambah ke sistem tanpa perlu me-*reboot* komputer. Ketika USB dipasang, ia langsung dikenal sistem komputer dan memproses *driver* yang diperlukan untuk menjalankannya.

Nomor kaki pada soket kabel USB A/B

Penetapan kaki

Kaki	Fungsi
1	V_{BUS} (4.75–5.25 V)
2	D−
3	D+
4	GND
Shell	Shield

Gambar 3. 3 dari kaki soket USB

Sedangkan untuk koneksi pada *game Immersive Tool* ini memakai USB A/B, hal ini bertujuan untuk mempermudah. Berikut ini spesifikasi dari koneksi USB A/B:

- Modus transfer*: Asinkron*
- Jumlah sistem yang ditemukan: 2-5
- Jumlah perangkat maksimal: 127
- IRQ: 11
- Panjang maksimal: 3-5 meter
- Maksimal data: 12 Mbit/sec (1.5 MB/sec)
- Power: 2.5w

3.1.2.1.1 Komunikasi Asinkron

Transmisi *Asinkron* memiliki kecepatan data yang bervariasi, oleh karena itu membutuhkan protokol/parameter-parameter yang harus dimengerti oleh peralatan yang ingin berkomunikasi. Sedangkan pada *Arduino Duemelanove* sudah terdapat protokol yang tersedia, jadi hasil data dari mikrokontroler yang berupa *String* akan diubah menjadi *bit.*

Protokol komunikasi data antara lain :

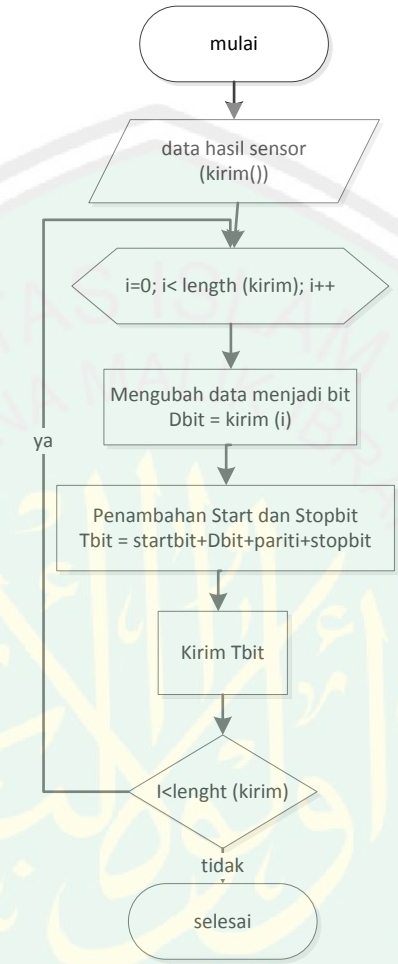
- *Start Bit* Selalu bernilai 0 : Ketika komunkasi UART *(serial asinkron)* akan diberikan, terlebih dahulu dimulai dengan pemberian *Start Bit*. Fungsinya sebagai pemicu (tanda) kepada penerima (RxD) bahwa akan ada data yang diberikan oleh pemancar (TxD) dan juga akan memicu *clock* pada *reciever* sehingga disinkronkan dengan *clock* pada *transmitter*. *Clock* penerima dan pemancar haruslah akurasi dengan toleransi 10% sehingga tidak terjadi kesalahan data.
- *Data Bits* : adalah data yang akan dikirimkan secara UART dimulai dari LSB (bit ke 0) hingga MSB (bit terakhir). Jangan lupa menentukan banyaknya bit tersebut haruslah sama antara pemancar dengan penerima. Banyaknya data bits pada AVR bisa bernilai 7,8 atau 9 data bit.
- *Parity*(keseimbangan) : berfungsi sebagai pengecekan *error* data yang ditransfer. *Parity* bisa bernilai ODD (ganjil), *EVEN* Genap), dan *NONE* selain itu pemancar dan penerima harus menggunakan *parity* yang sama. Jika ODD *parity* maka jumlah total nilai 1 pada data *bit* + parity berjumlah ganjil,

contoh ODD, jika data bits 00110101 maka *parity* bernilai 1. Sedangkan jika EVEN *parity* maka jumlah total nilai 1 pada data *bit* + *parity* berjumlah genap. Contoh *even*, jika data bits 00110101 maka *parity* bernilai 0.

- *Stop Bit* Selalu bernilai 1 : berfungsi sebagai akhir dari komunikasi data dan kamudian masuk pada *IDLE state*. Pengiriman data selanjutnya dapat dilakukan setelah *stop bit* diberikan.
- *IDLE state* : adalah kondisi tidak terjadinya komunikasi data dan jalur data berlogika 1 secara terus menerus *(marking)*.

Sedangkan format pengiriman data untuk arduino memakai **9600-8-N-1**, dimana 9600 merupakan *baud rate*, 8 merupakan jumlah data, N *pariti* dan 1 merupakan *stop bit*. Jadi data dari mikrokontroler yang berupa *string* akan dikonversikan berupa data bit, dan data bit ini yang akan dikirim melalui tranmisi *asinkron*. Dialam pengiriman data terdapat 1 karakter yang merupakan 1 *bit* dan menjadi 8 *bit*. Dalam pengiriman data pasti ada pengirim (TX) dan penerima (RX), yang digambarkan pada *flowchart* dibawah.

Proses pengiriman data serial *asinkron*

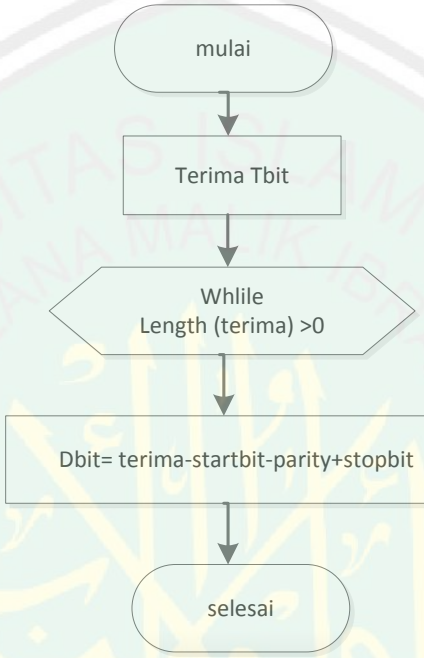


Gambar 3. 4 gambar proses pengiriman data serial *asinkron*

Dari *flowchart* di atas dapat diterangkan bahwa data dari *string* dari setiap hasil sensor dikirim ke PC melalui USB, dengan mengubah data dari hasil mikrokontroler menjadi bit dengan protokol dan data tersebut dikirim dari jumlah data pertama sampai akir dalam satu *blok*(i=0;i*length*(kirim);i++). Jika data dalam satu *blok* sudah menjadi bit maka format pengiriman *asinkron* dengan menambahi *start bit*(0)+databit+*pariti*+*stopbit*(1), dan kemudian data *bit* akan dikirim. Dan

jika terjadi kesalahan maka data bit akan dikirim lagi untuk diproses lagi tiap blok.

Proses penerima data serial *asinkron*

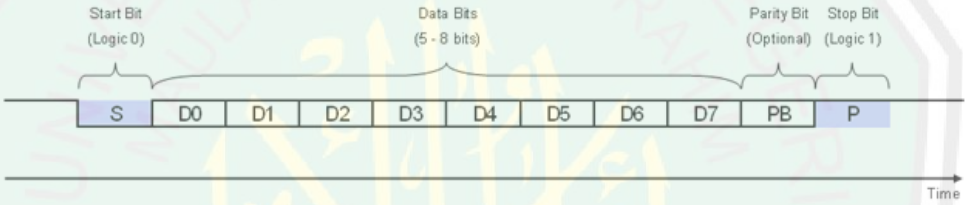


Gambar 3. 5 gambar proses penerima data serial *asinkron*

Setelah data bit diterima maka diproses dengan perulangan, untuk menerima dari kiriman data dari yang pertama (*length*(terima)>0), ketika data sudah diterima dalam satu blok maka data yang dikirim akan dikurangi *startbit*, *parity* dan *stopbit*. Maka akan muncul hasil dari data bit dari awal data bit kirim dari mikrokontroler.

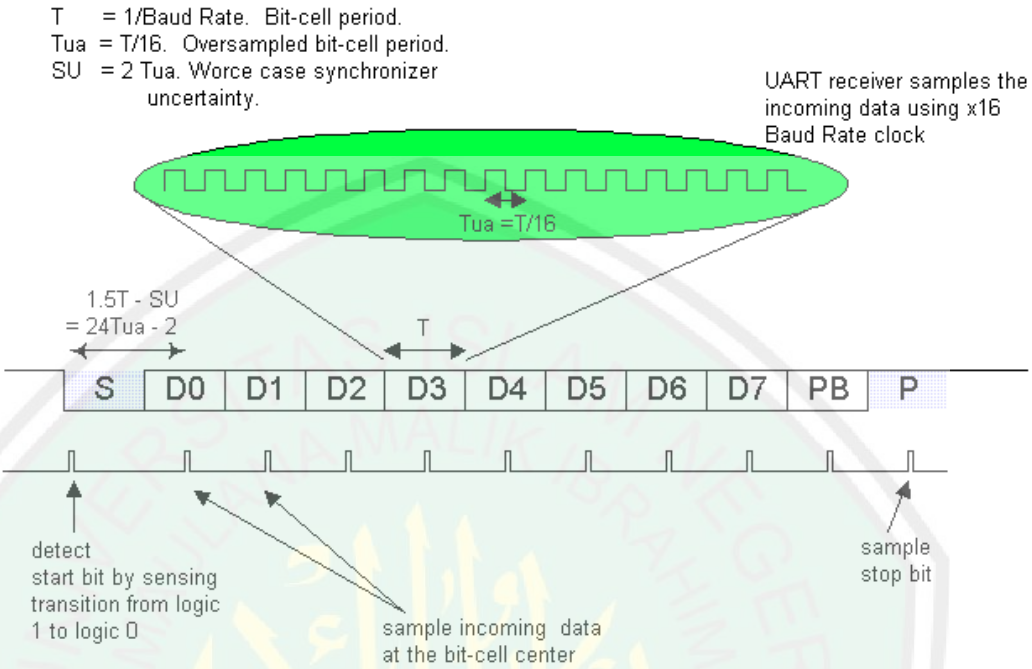
The Universal Asynchronous Receiver / Transmitter (UART) *controller* adalah komponen kunci dari komunikasi serial subsistem dari sebuah komputer dan pada mikrokontroler *Arduino Duemelanove* memakai UART. UART juga fitur terintegrasi umum di kebanyakan mikrokontroler. Transmisi *Asinkron*

memungkinkan data yang akan dikirimkan tanpa pengirim harus mengirim sinyal *clock* ke penerima. Dalam hal ini, pengirim dan penerima harus setuju pada parameter waktu *(Baud Rate)* transmisi sebelum dan bit khusus ditambahkan untuk setiap kata untuk menyinkronkan unit pengirim dan penerima. Dalam transmisi *asinkron*, pengirim mengirimkan *bit Start*, 5 sampai 8 bit data (LSB pertama), bit paritas *opsional*, dan kemudian 1, 1,5 atau 2 Berhenti bit (kong, 2010).



Gambar 3. 6 dasar paket UART: 1 Mulai bit, 8 bit data, 1 bit paritas dan 1 bit berhenti

Setiap operasi perangkat keras UART dikendalikan oleh sinyal *clock* yang berjalan pada tingkat yang jauh lebih cepat daripada tingkat *baud rate*. Untuk mencari frekuensi sampling dari pengiriman serial menggunakan UART ini dengan rumus *baud rate* di kalikan 16, dimana *baud rate* 9600 x 16 = 153600. Jadi frekuensi dari pengiriman data serial ini 153600.



Gambar 3. 7 data titik pengambilan sampel oleh penerima *UART*

Transmisi dan menerima *UART* harus ditetapkan pada tingkat yang sama *baud*, panjang karakter, *paritas*, dan *stop bit* untuk operasi yang tepat. Format khas untuk *port* serial digunakan dengan PC yang terhubung ke mikrokontroler adalah 1 Mulai bit, 8 bit data, tidak ada paritas dan 1 Berhenti bit. *UART* adalah bentuk sederhana dari komunikasi antara mikrokontroler dan PC. Namun, karena pertumbuhan yang menjamur teknologi, *port* serial secara perlahan digantikan dengan cara lain *port* komunikasi. Namun demikian, komunikasi serial masih mungkin bahkan tanpa *port* serial fisik pada PC Anda yang menggunakan USB (*universal serial bus*) (Teguh, 2003).

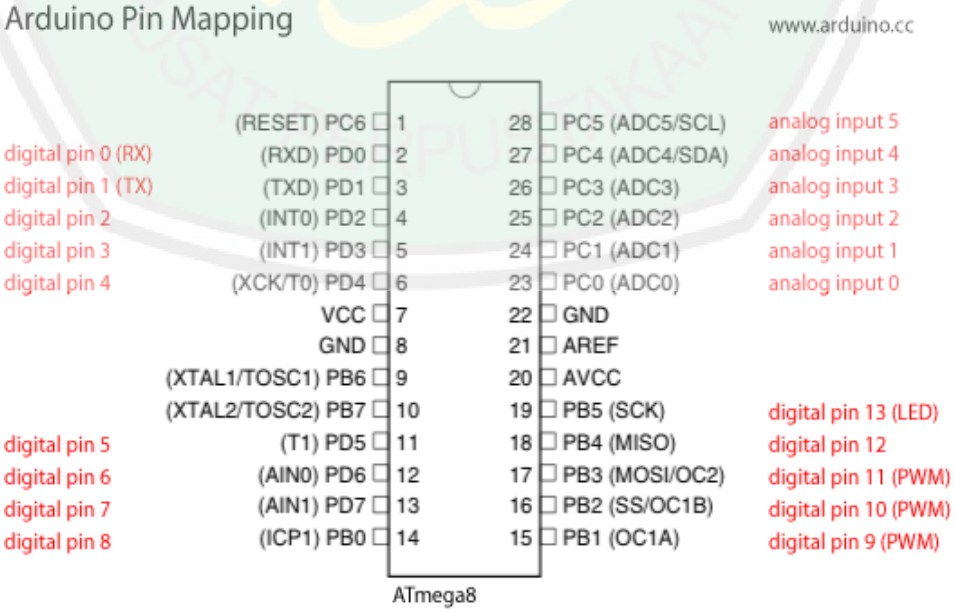
❖ Definisi metode *Asinkron* sebagai berikut:

- Jumlah bit tiap karakter adalah 5 sampai 8 bit.
- *Parity* bit yang digunakan untuk mendeteksi kesalahan *(error)* yang berbentuk *odd* (ganjil), *even* (genap) atau tanpa *parity (no parity).*
- Jumlah *stop bit* (1 bit, 1,5 bit, atau 2 bit) dan 1 *start bit*.
- *Baud rate* atau kecepatan data (bps).

3.1.2.2 Mikrokontroler *Arduino Duemilanove*

Duemilanove adalah mikrokontroler CMOS 8-bit berarsitektur AVR RISC yang memiliki 8K *Bytes In-System Programmable Flash*. Mikrokontroler dengan konsumsi daya rendah ini mampu mengeksekusi instruksi dengan kecepatan maksimum 16 MIPS pada frekuensi 16MHz.

Konfigurasi Kaki – kaki Mikrokontroler *Duemilanove*



Gambar 3. 8 komfigurasi kaki-kaki *Arduino*

Mikrokontroler *Duemilanove* mempunyai dua *timer* yaitu *TIMER 0* dan *TIMER 1*. Kedua *timer* saling berbagi dua macam SFR *(Special Function Register)* yang mengontrol *timer* masing – masing *timer* memiliki dua macam SFR yang spesifik, yaitu TH0/TL0, untuk *timer* 0 dan TH1/TL1 untuk *timer* 1.

3.1.2.2.1 ADC

ADC adalah singkatan dari *Analog to Digital Converter*. Fungsinya merubah data analog menjadi digital sehingga dapat diproses oleh *mikrokontroler*. Hal-hal yang juga perlu diperhatikan dalam penggunaan ADC adalah tegangan maksimum yang dapat dikonversikan oleh ADC dari rangkaian pengkondisi sinyal, resolusi, pewaktu eksternal ADC, tipe keluaran, ketepatan dan waktu konversinya.

Beberapa karakteristik penting ADC:

1. Waktu konversi
2. Resolusi
3. Keetidak linieran
4. Akurasi

Ada banyak cara yang dapat digunakan untuk mengubah sinyal analog menjadi sinyal digital yang nilainya proposional. Jenis ADC yang biasa digunakan dalam perancangan adalah jenis *successive approximation convertion* atau pendekatan bertingkat yang memiliki waktu konversi jauh lebih singkat dan tidak tergantung pada nilai masukan analognya atau sinyal yang akan diubah.

3.1.2.2.2 Prosedur Pengisian Program Mikrokontroler menggunakan *Arduino development environment*

Adapun prosedur atau tata cara pengisian program ke mikrokontroler *Duemilanove* adalah sebagai berikut :

- Program dibuat dalam bahasa C yang ditulis di *Code Vision* AVR, namun bisa juga diketik dengan mengunakan editor yang mendukung dalam pengisian program pada mikrokontroler *Arduino Duemilanove*.
- Pada saat penyimpanan program yang dibuat sebaiknya disimpan pada satu *folder*, tujuannya agar pada saat membutuhkan program yang telah dibuat bisa di *download*.
- Program yang telah disimpan harus di komplikasi *(compiler)*, tujuannya agar program yang kita simpan menjadi bahasa mesin sehingga dimengerti oleh mikrokontroler, dengan menekan tombol F9.
- Sebelum melakukan kompilasi terlebih dahulu melakukan pengecekan terhadap *error* pada program yang dibuat.
- Untuk mulai mengisi atau *mendownload* program ke mikrokontroler langsung dengan *Code Vision* AVR, dengan cara menekan Ctrl+F9 maka tampil dilayar proses pengisian *programing* pada mikrokontroler tunggu sampai pengisian 100%.

Bahasa pemrograman *Arduino* adalah bahasa C. Tetapi bahasa ini sudah dipermudah menggunakan fungsi-fungsi yang sederhana sehingga pemula pun bisa mempelajarinya dengan cukup mudah. Untuk membuat program *Arduino* dan

mengupload ke dalam *board Arduino*, anda membutuhkan *software Arduino IDE (Integrated Development Enviroment)*.

Gambar 3. 9 *Integrated Development Enviroment*

Berikut ini adalah tombol-tombol *toolbar* serta fungsinya:

Verify untuk mengecek *error* pada *code* program.

Upload untuk meng *compile* dan meng *upload* program ke *Arduino board*.

New untuk membuat *sketch* baru.


Open untuk menampilkan sebuah menu dari seluruh *sketch* yang berada di dalam.


sketchbook untuk save menyimpan *sketch*.


Serial Monitor untuk membuka serial monitor.

3.1.2.3 Sensor

3.1.2.3.1 Sensor kecepatan menggunakan LED infra merah dan Photo Dioda.

LED Infra Merah merupakan salah satu jenis LED (*Light Emiting Diode*) yang dapat memancarkan cahaya infra merah yang tidak kasat mata. Cahaya infra merah merupakan gelombang cayaha yang berapa pada *spektrum* cahaya tak kasat mata. LED infra merah dpat memacarkan cahaya infra merah pada saat dioda LED ini diberikan tegangan bias maju pada anoda dan katodanya. LED infra merah ini dapat memancarkan gelombang cahaya infra merah karena dibuat dengan bahan khusus untuk memendarkan cahaya infra merah. Secara teoritis LED infra merah mempuyai panjang gelombang 7800 Å dan mempuyai daerah frekuensi 3.104 sampai 4.104 Hz. Dilihat dari jangkah frekuensi yang begitu lebar, infra merah sangat fleksibel dalam pengunaanya.

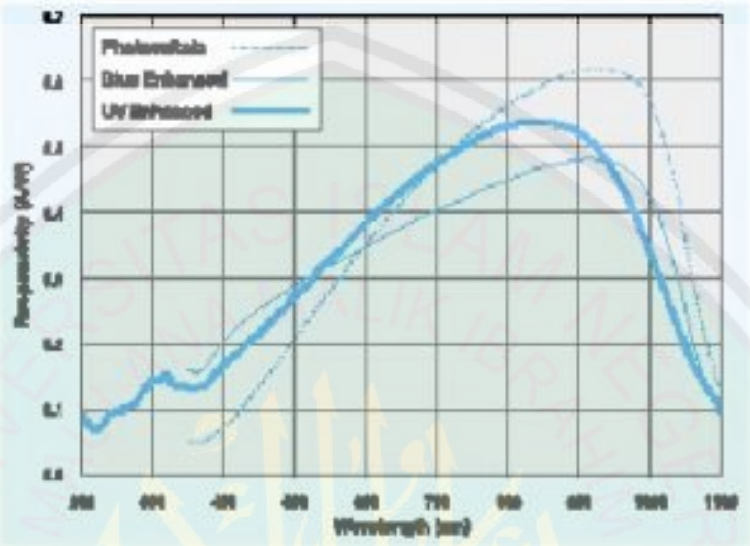
Gambar 3. 10 bentuk LED Infra merah

Cahaya infra merah tidak mudah terkontaminasi dengan cahaya lain, sehingga dapat digunakan baik siang maupun malam. Aplikasi cahaya infra merah sendiri dapat digunakan sebagai *link* pada jaringan telekomunikasi atau dapat juga dipancarkan pada *fiber optic*. Sebagai *receiver* cahaya infra merah dapat digunakan *photo dioda*, maupun modul *receiver* infra merah.

LED Photo dioda merupakan dioda yang peka terhadap cahaya, sensor *photo dioda* akan mengalami perubahan resistansi pada saat menerima intensitas cahaya dan akan mengalirkan arus listrik secara *forward* sebagaimana dioda pada umumnya. Sensor *photo dioda* adalah salah satu jenis sensor peka cahaya *(photodetector)*. Jenis sensor peka cahaya lain yang sering digunakan adalah *phototransistor*.

Tanggapan *frekuensi* sensor *photo dioda* tidak luas. Dari rentang tanggapan itu, sensor *photo dioda* memiliki tanggapan paling baik terhadap

cahaya infra merah, tepatnya pada cahaya dengan panjang gelombang sekitar 0,9 µm. Kurva tanggapan sensor *photo dioda* ditunjukkan pada gambar berikut:

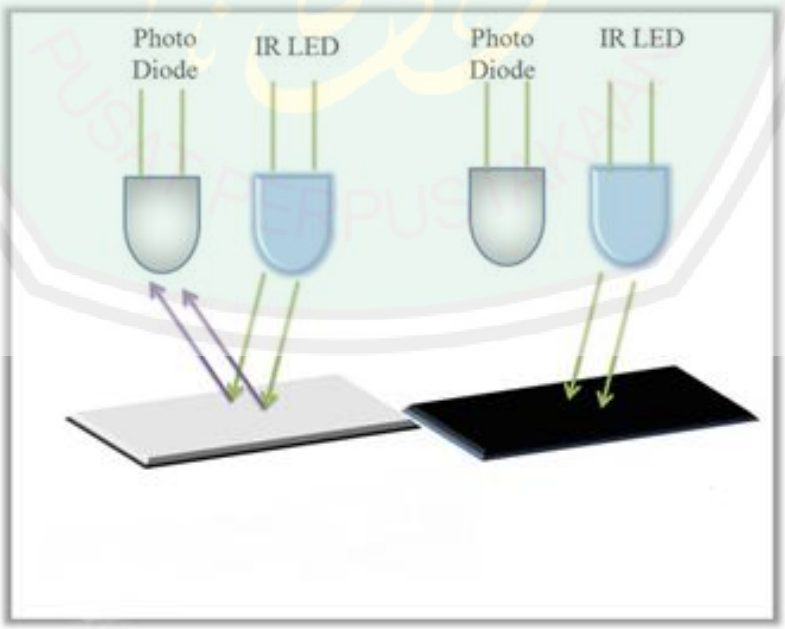


Gambar 3. 11 kurva tanggapan *Frekuensi* Sensor *Photo dioda*

Hubungan antara keluaran sensor *photo dioda* dengan intensitas cahaya yang diterimanya ketika dipanjar mundur adalah membentuk suatu fungsi yang *linier*. Prinsip kerja *photo dioda* ketika sebuah *photon* (satu satuan energi dalam cahaya) dari sumber cahaya diserap, hal tersebut membangkitkan suatu *elektron* dan menghasilkan sepasang pembawa muatan tunggal, sebuah *elektron* dan sebuah *hole*, di mana suatu *hole* adalah bagian dari kisi-kisi *semi konduktor* yang kehilangan *elektron.*

❖ Cara kerja Infra merah dan *Photo Dioda*

1. Prinsip kerja dari rangkaian ini adalah *photo dioda* akan ditembak secara terus menerus oleh cahaya infra merah, apabila nanti akan mendeteksi warna hitam dan putih. Untuk nilai hitam akan muncul 0 sedangkan warna putih akan mempunyai nilai 1.
2. Proses penghitungannya dilakukan dengan mendeteksi adanya perpotongan pada jalur infra merah.
3. Setiap perpotongan akan memberikan perubahan kondisi logika dari 0 ke 1 selama selang waktu tertentu.
4. Perubahan kondisi logika ini yang digunakan sebagai acuan perhitungan.
5. Sensor ini diletakkan pada jalur yang akan dilewati garis hitam maupun garis putih.



Gambar 3. 12 proses sensor saat mendeteksi warna hitam dan putih

Gambar 3. 13 penempatan sensor kecepatan di sepeda

Sedangkan nantinya data dari sensor akan diproses mikrokontroler menjadi RPS, data ini yang akan dikirim ke PC. Sedangkan penghitungan kecapatan akan diperoleh dari berapa kali sensor membaca titik hitam (0) dan titik putih (1) dalam tiap detik. Maka dari itu akan diketahui kecepatan tiap detik, maka data yang yang ditampilkan di *speedometer* nanti merupakan data kecepatan rotasi roda, yang diperoleh dari hasil data sensor kecepatan.

3.1.2.3.2 Sensor kemudi dan pengereman

Untuk sensor kemudi dan pengereman ini menggunakan *Potensiometer*. ***Potensiometer*** adalah resistor tiga terminal dengan sambungan geser yang

membentuk pembagi tegangan dapat disetel. Jika hanya dua terminal yang digunakan (salah satu terminal tetap dan terminal geser), *potensiometer* berperan sebagai *resistor variabel* atau *Rheostat*. *Potensiometer* biasanya digunakan untuk mengendalikan peranti elektronik seperti pengendali suara pada penguat.

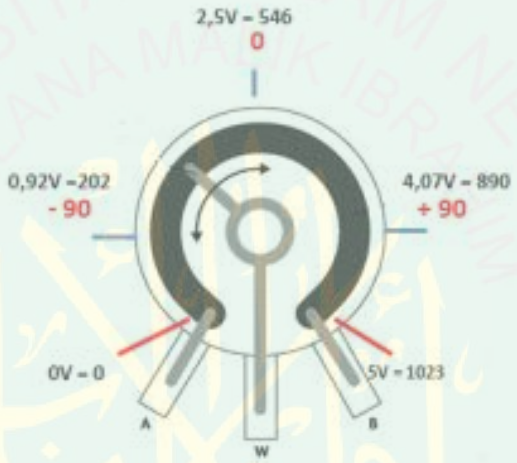
Gambar 3. 14 dari penempatan potensio unutk sensor kemudi dan pengereman

Dari Segi mekanik, *potensiometer* dapat diletakkan pada posisi yang kita inginkan karena dilihat dari bentuknya yang *simple* dan juga mendukung *mekanik*. Selain itu juga ada banyak pilihan bentuk *potensiometer* yang tersedia di pasaran. *Potensiometer* yang tersedia di pasaran terdiri dari beberapa jenis, yaitu: *potensiometer* karbon, *potensiometer wire wound* dan *potensiometer metal film*.

Dari Segi elektrik , penggunaan *potensiometer* sebagai sensor posisi cukup praktis karena hanya membutuhkan satu tegangan eksitasi dan biasanya tidak membutuhkan pengolahan sinyal yang rumit. Dari segi *programming*, perubahan

posisi dapat diukur dari perubahan resistansi yang dimiliki *potensiometer* yang sebelumnya telah dikonversi menjadi sinyal inputan yang sesuai dengan kontroler baik tegangan maupun arus.

Berikut ini adalah gambaran dari sensor *potensiometer* yang digunakan untuk sensor kemudi.



Gambar 3. 15 *potensiometer* dengan belokan kanan dan kiri

Pada gambar ditunjukan bahwa pada posisi 0°(2,5V-546) maka posisi kemudi tepat lurus, sedangkan pada posisi -90°(0,92V-202) maka kemudi belok kiri dan 90°(4,7V-890) akan belok kanan. Maka dari itu untuk belok kiri nanti dari posisi 0 tegangan akan semakin rendah, begitu juga sebaliknya jika belok kanan maka tegangan akan tambah besar dari posisi 0 dan daya yang digunakan adalah 5V dan 10bit(0-1023).

Sertiap hasil data dari sensor nantinya akan diolah pada mikrokontroler dan data akan dihitung dan dijadikan sudut belokan dari paling kiri (-90) dan

paling kanan (90) sedangkan nilai tengah (0). Yang nantinya data hasil yang akan dikirim ke PC dan akan dimasukkan pada *game*.

Dalam sensor pengereman jika kita mengerem nantinya pasti akan mempengarui tentang kecepatan, pengereman dilakukan ketika *player* ingin berbelok maupun terlalu kencang dalam mengayuh. Selaian pengereman dengan sensor potensio sendiri dalam sepeda ini juga masih terdapat rem yang masih berfungsi, hal ini diupayakan untuk seperti sepeda pada umumnya.

3.1.2.3.3 Tombol Push On

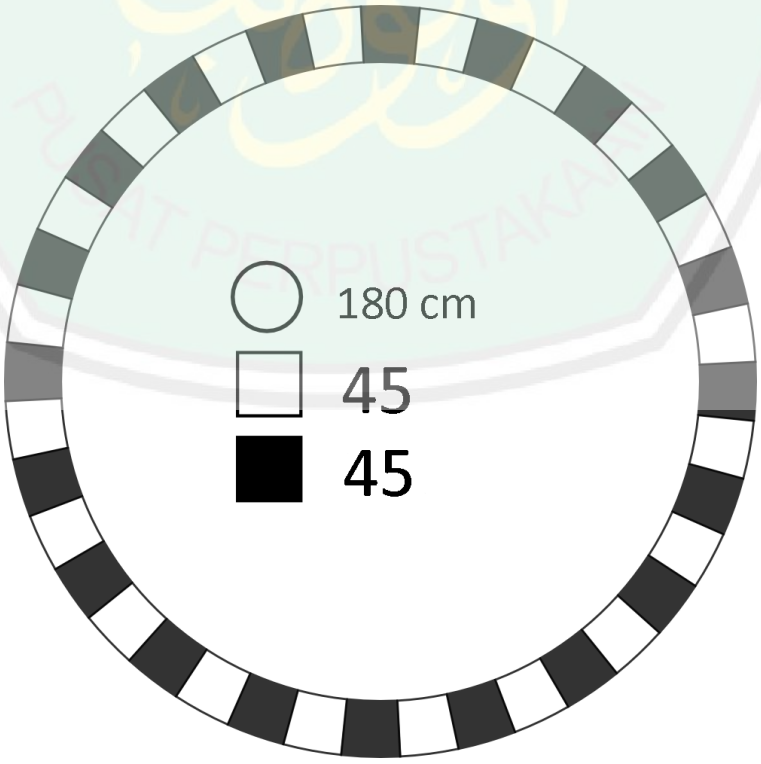
Saklar merupakan komponen elektronika yang berfungsi untuk menghubungkan dan memutuskan dua titik atau lebih dalam suatu rangkaian elektronika. Salah satu jenis saklar adalah saklar *Push ON* yaitu saklar yang hanya akan menghubungkan dua titik atau lebih pada saat tombolnya ditekan dan pada saat tombolnya tidak ditekan maka akan memutuskan dua titik atau lebih dalam suatu rangkaian elektronika dan saklar ini terhubung langusng pada *port* arduino A2. Simbol saklar *Push ON* ditunjukan pada gambar berikut.

Gambar 3. 16 tombol *push on*

Gambar di atas merupakan gambaran dari tombol *push on*, tombol model ini dirasa cocok dari pada tipe tombol lainnya. *Push on* ini berfungsi untuk mundur, jadi ketika *player* ingin mundur karena menabrak atau keliru berbelok hanya cukup menekan tombol yang disiapkan pada setang kemudi sebelah kiri.

3.1.2.4 Statistik Data

Statistik data digunakan untuk mancari nilai kecepatan rata, kecepatan maksimal pada pada *game* yang menggunakan *Immersive Tool*. Data kecepatan diperoleh dari berapa kali sensor membaca titik putih (1) dan titik hitam (0) pada roda, dan data akan dikirmkan ke mikrokontroler untuk diproses. Dalam mikrokontroler nantinya diberikan rumus menghitung kecepatan, kecepatan rata-rata, kecepatan maksimal.



Gambar 3. 17 tanda sensor pada roda

Dari gambar tanda roda di atas dapat diketahui bahwa untuk keliling roda 180 cm, sedangkan tanda putih berjumlah 45 dan tanda hitam berjumlah 45. Untuk perberian tanda ini diusahakan genap dan benar-benar presisi, karena jika terdapat tanda sensor yang tidak sama maka pembacaan sensor tidak *valid*. Hasil perhitungan data nantinya akan dapat ditampilkan pada *speedometer*, sedangkan untuk mencari data kecepatan dapat menggunakan rumus.

$$speed = \frac{count\ A}{jumlah\ tanda} \times 1.000 \times waktu\ sampling$$

Di mana:

Count A: jumlah baca tanda tiap detik

Jumlah tanda: jumlah data pada roda

1.000: waktu sampling menggunakan detik

Misal: $\frac{45}{45} \times 1.000 \times 1.000 = 1$ rps

Hasil data dari perhitungan di atas akan berupa dalam satuan detik. Jadi berapa kali sensor membaca jumlah warna putih tiap detik, maka hasil ini yang akan dihitung. Sedangkan untuk menghitung kecepatan rata-rata dengan rumus perhitungan.

$$AVG = \frac{kecepatan\ 1 + \cdots + kecepatan\ N}{N}.$$

Dimana:

AVG (*Average*) kecepatan rata

Kecepatan 1 – kecepatan N ialah hasil kecepatan perdetik

N merupakan jumlah dari dari data

Misal: Misal: $AVG = \frac{1+2+5+4+3}{5} = \frac{15}{5} = 3\ rps$

Sedangkan untuk menghitung nilai rata-rata per detik akan menjumlahkan semua data hasil per detik, berikut ini permisalan unutk gambaran mencari kecepatan rata-rata.

$= \frac{1}{1} = 1$

$= \frac{1+2}{2} = 1{,}5$

$= \frac{1+2+5}{3} = 2{,}6$

$= \frac{1+2+5+4}{4} = 3$

$= \frac{1+2+5+4+3}{5} = 3$

} nilai rata-rata

Sedangkan untuk mencari kecepatan maksimal akan mancari nilai hasil tertinggi tiap detik, dalam perhitungan kecepatan maksimal nantinya yang akan ditampilkan pada hasil *speedometer* yang akan diinialisasikan dengan.

```
max=0
        for(int i=1; i<=5; i++)
         {
                if(V(i)>max)
                {
                        Max= V(i)
                }
        }
```

Potongan kode di atas adalah mencari kecepatan maksimal

Gambar dari koding mikrokontroler untuk mencari kecepatan maksimal.

Sebagai permisalan

If (V(1)>0) = 1
If (V(2)>1) = 2
If (V(5)>2) = 5
If (V(4)>5) = 5
If (V(3)>5) = 5
} maka kecepatan maksimal adalah 5

Jadi dari gambaran di atas dapat diketahui bahwa kecepatan maksimal dari seluruh data hasil yaitu 5. Jika sudah didapat nilai per detik dari sensor kecepatan nantinya akan mencari kecepatan jarak tempuh, menggunakan rumus.

$$kecepatan = rps \times keliling\ roda$$

Jadi dapat dihitung bahwa Kec: 3 x 1,5 = 4,5 m/s

3.1.3 Kalibrasi Sensor

Kalibrasi adalah kegiatan untuk menentukan kebenaran konvensional nilai penunjukkan alat ukur dan bahan ukur dengan cara membandingkan terhadap standar ukur yang mampu telusur *(traceable)* ke standar nasional maupun internasional untuk satuan ukuran dan bahan-bahan acuan tersertifikasi.

Dalam setiap sensor yang dipakai pasti ada penyusutan nilai keakuratan deteksi. Banyak cara untuk kalibrasi, salah satunya juga bisa dengan *Arduino*

Development Environment. Nantinya kita bisa melihat nilai yang terdapat pada program, misal pada sensor potensio Sangatlah sulit untuk mencari nilai 0, maka dari itu dicari nilai yang paling mendekati nilai 0. Selain memakai *Arduino Development Environment* juga bisa memakai *Voltmeter*, cara ini sangat manual sehingga dicari nilai tengah dari seluruh nilainya.

Berikut ini adalah garis besar dari kalibrasi:

❖ **Tujuan Kalibrasi**

- Mencapai ketertelusuran pengukuran. Hasil pengukuran dapat dikaitkan/ditelusur sampai ke *standar* yang lebih tinggi/teliti *(standar primer nasional* dan / *internasional)*, melalui rangkaian perbandingan yang tak terputus.
- Menentukan *deviasi* (penyimpangan) kebenaran nilai *konvensional* penunjukan suatu *instrument* ukur.
- Menjamin hasil-hsil pengukuran sesuai dengan standar nasional maupun internasional.

❖ **Manfaat Kalibrasi**

- Menjaga kondisi instrumen ukur dan bahan ukur agar tetap sesuai dengan spesifikasinya.
- Untuk mendukung sistem mutu yang diterapkan di berbagai industri pada peralatan laboratorium dan produksi yang dimiliki.
- Bisa mengetahui perbedaan (penyimpangan) antara harga benar dengan harga yang ditunjukkan oleh alat ukur.

❖ **Prinsip Dasar Kalibrasi**

- Obyek Ukur *(Unit Under Test).*
- Standar Ukur (Alat standar kalibrasi, *Prosedur/Metrode* standar (Mengacu ke standar kalibrasi internasional atau prosedur yg dikembangkan sendiri oleh laboratorium yg sudah teruji *(diverifikasi*)).
- Operator/Teknisi (dipersyaratkan operator/teknisi yg mempunyai kemampuan teknis kalibrasi (bersertifikat)).
- Lingkungan yg dikondisikan (Suhu dan kelembaban selalu dikontrol, gangguan faktor lingkungan luar selalu diminimalkan & sumber ketidak pastian pengukuran).

❖ **Kalibrasi Diperlukan Untuk**

- Perangkat baru.
- Suatu perangkat setiap waktu tertentu.
- Suatu perangkat setiap waktu penggunaan tertentu (jam operasi).
- Ketika suatu perangkat mengalami tumbukan atau getaran yang berpotensi mengubah kalibrasi.
- Ketika hasil pengamatan dipertanyakan.

Kalibrasi semua sensor yang digunakan menggunakan *Arduino Development Environment*, sebelum kalibrasi dijalankan nantinya pada mikrokontroler sudah diberikan koding yang dimana bisa melihat data yang akan dimunculkan pada tampilan *Arduino Development Environment*. Jika sudah

terkoneksi antara sensor, mikrokontroler dan PC nantinya data akan dicari titik yang diinginkan untuk memulai dan akir.

3.1.3.1 Kalibrasi Sensor Kecepatan

Kalibrasi sensor kecepatan menggunakan *Arduino Development Environment*. Rangkaian sensor kecepatan yang berupa LED infra merah dan *photo dioda* akan dihubungan pada mikrokontroler, dalam mikrokontroler akan diberikan koding tentang kalibrasi kecepatan. Pada *Arduino Deveopment Environment* nanti akan *diupload* koding yang sudah jadi, selanjutnya rangkaian sensor kecepatan akan dibuhungan pada mikrokontroler dan penghubungkan mikrokontroler pada PC menggunakan USB. Berikut koding yang ter-*upload* pada mikrokontroler untuk kalibrasi sensor kecepatan.

```
void hitungKecepatan()
{
   static unsigned int countBaling,countBalingtest;
   static unsigned long lastTime;
   static boolean statKec1, statKec2;
void hitungKecepatan()
{
   static unsigned int countBaling,countBalingtest;
   static unsigned long lastTime;
   static boolean statKec1, statKec2;

//countBalingtest hanya digunakan untuk kalibrasi jumlah pulsa 1
putaran penuh.
  //jika sudah dikalibrasi tidak perlu di kirim ke serial
  unsigned int analogKec=analogRead(A1);
  if(analogKec<batasBlackWhite) //analog dijadikan digital
  {
        statKec1=HIGH;
      digitalWrite(13,  HIGH);  //menyalakan  led  internal  arduino
agar terlihat beda sense hitam putih
  }
  else
  {
        statKec1=LOW;
        digitalWrite(13, LOW);
  }
```

```
    //bila sama maka sensor masih menghitung salah satu hitam atau
putih dan belum berubah

  if(statKec2!=statKec1)
  {
      if( statKec1==HIGH)//rising .. dari lowh menjadi high yg
dihitung. kalau high jadi low tidak dihitung.
      {
            countBaling++;
            countBalingtest++;
      }
    statKec2=statKec1;
  }

  if(millis()-lastTime>= waktuSampling )/*waktuSampling diganti
dengan 1000 = 1 detik*/
  {
      //rps = rotation per second.
    double
rps=((double)countBaling/jmlBaling)*(1000.0/waktuSampling);
      double kecepatanMperSecond = rps * kelilingRoda;

      //timer setelah hitung kecepatan putar selama >= waktu
sampling lalu kirim
      //derajat kemudi # rem # kecepatanMperSecond # mundur   #
countBaling # countBalingtest

      //countBaling = jumlah pulsa yang dihasilkan baling dalam
waktu sampling
      //countBalingtest = jumlah pulsa mulai dari hidup

      //mode pengiriman data
      if(mode=='D')//debug mode
      {
          unsigned long runningTime = millis();
            kirim                                                =
(String)derajatKemudi()+"#"+Rem()+"#"+kecepatanMperSecond+"#"+digi
talRead(A2)+"#"+rps+"#"+countBaling+"#"+countBalingtest+"#"+(runni
ngTime/1000)/3600+" hours "+((runningTime/1000)/60)%60 +" minutes
"+(runningTime/1000)%60+" seconds" ;

      }
      else if(mode=='R')//running mode
      {
            //kirim                                              =
(String)derajatKemudi()+"#"+Rem()+"#"+kecepatanMperSecond+"#"+digi
talRead(A2);
                  kirim = (String)derajatKemudi();
      }
      else
      {
            kirim = "info#Please set send mode fisrt";
      }

    Serial.println(kirim);
      countBaling = 0;
```

```
        lastTime=millis();
    }

}
```

Potongan kode di atas adalah kalibrasi kecepatan

Keterangan program di atas adalah koding yang sudah di *upload* didalam mikrokontroler, setalah data di *upload* maka untuk langkah selangkah selanjutnya tinggal menyambungkan rangkaian sensor ke mikrokontroler dan langsung disambungkan ke PC. Maka akan diketahui kalibrasi data dari sensor tersebut.

3.1.3.2 Kalibrasi Sensor Kemudi dan Rem

Untuk proses kalibrasi sensor kemudi dan rem yang menggunakan *potensiometer* ini sama, sama-sama menggunakan *Arduino Development Environment.* Yang membedakan cara kalibrasi, berikut ini adalah cara kalibrasi sensor kemudi dan rem menggunakan *potensiometer*.

❖ Kalibrasi Sensor Kemudi

Kalibrasi sensor kemudi dengan cara memberikan koding pada mikrokontroler dan tinggal manyambungkan antara sensor ke mikrokontroler lalu tinggal kita lihat pada *Arduino Development Environment* yang sudah ter*instal* pada komputer. Dibawah ini koding yang terdapat pada mikrokontroler yang terdapat untuk kalibrasi sensor kemudi.

```
double derajatKemudi()
{
      //range setir -90 s/d 90 = 180
      int valueIn = analogRead(A0);
      double derajat=-90 +(valueIn-lowIn)*(180.00/(highIn-lowIn));
      //pembacaan akan eror nilainya jika derajat setir lebih dari
batas yang ditentukan -90 sampai 90
      //di paskan
```

```
        if(valueIn<lowIn)
              derajat = -90.00;
        else if(valueIn>highIn)
              derajat = 90.00;


        return derajat;
}
```

Potongan kode di atas adalah kalibrasi sensor kemudi

❖ Kalibrasi Sensor Rem

Untuk kalibrasi sensor hampir sama tahapannya dengan sensor kemudi, cuman kodingnya dan perumusan yang membedakan. Berikut ini adalah koding dari kalibrasi sensor rem.

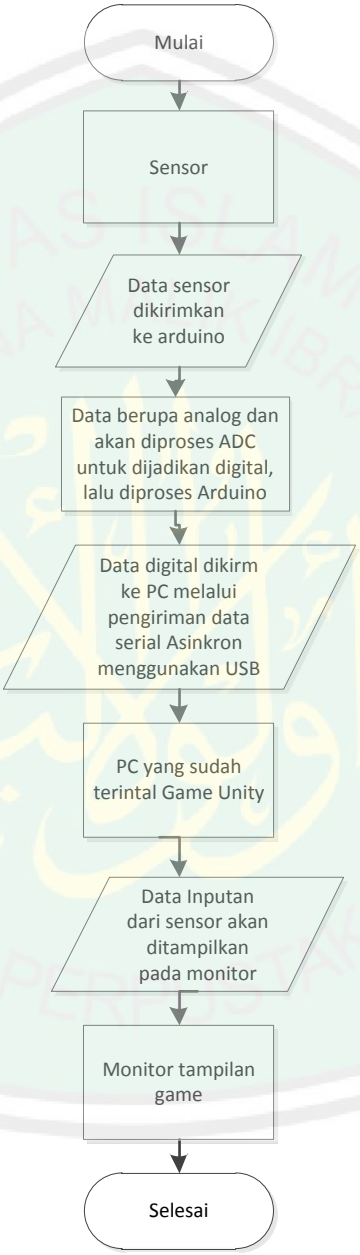
```
double Rem()
{
        //range rem 0-9.
        int valueIn = analogRead(A3);
        double rem=0.0 +(valueIn-lowRem)*(9.00/(highRem-lowRem));
        if(valueIn<lowRem)
              rem = 0.00;
        else if(valueIn>highRem)
              rem = 9.00;

        return rem;
}
```

Potongan kode di atas adalah kalibrasi rem

3.2 Perancangan Alur Proses

3.2.1 Perancangan alur proses pengiriman data multisensor

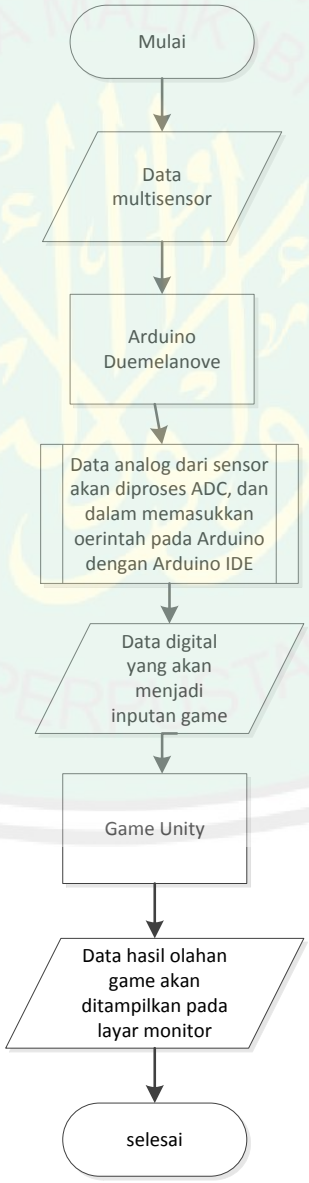


Gambar 3. 18 rancangan alur proses sensor data

Dari gambar di atas dapat dijelaskan semua sensor akan mengirimkan data ke mikrokontroler, pada mikrokontroler data tersebut akan diolah menjadi data

digital dan dikirim menggunakan pengiriman data serial *asinkron* menggunakan USB. Setelah data terdeteksi sebagai inputan *game* maka data akan diproses pada *game* dan pada *game* akan ditampilkan pada layar permainan.

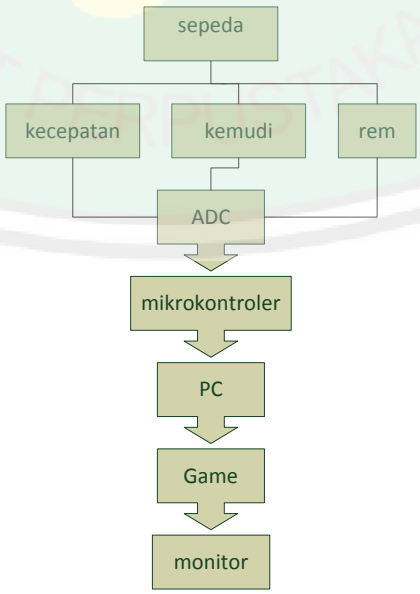
3.2.2 Proses pengolahan data *multisensor*



Gambar 3. 19 proses pengolahan data *multisensor*

Dari gambar proses di atas dapat diketahui bahwa data *multisensor* yang masih analog akan diubah ADC untuk dijadikan data digital sebelum diolah pada mikrokontroler. Setelah data dari *multisensor* menjadi digital maka akan diolah pada mikrokontroler yang dimana dalam mikrokontroler sudah di *upload* koding menggunakan *Arduino Development Environment*, termasuk proses kalibrasi. Data yang sudah diproses oleh mikrokontroler akan dikirim ke PC menggunakan komunikasi serial *asinkron* yang menguhungkan dengan USB. Pada komputer nantinya akan diatur supaya bisa koneksi dengan mikrokontroler, setelah data bisa tersambung maka langkah selanjutnya memberikan perintah pada koding *game* untuk memanggil data *multisensor* yang dikirim untuk dijadikan inputan *game*. Jika dalam *game* sudah selesai maka *player* sudah bisa bermaian *game* sepeda yang menggunakan *Immersive Tool* dengan tampilan di monitor.

3.2.3 Blok Diagram



Gambar 3. 20 Diagram Blok

Dari diagram blok diatas dapat dijelaskan bahwa *immersive tool* ini nantinya sebagai alat kontrol simulasi, yang mana nantinya terdapat sensor kemudi, sensor rem, sensor kecepatan dan juga tombol *push on*. Data hasil sensor ini akan diolah pada mikrokontroler, yang mana sebelum diolah mikrokontroler nantinya data hasil sensor yang masih analog akan diubah menjadi digital oleh ADC. Hasil data dari mikrokontroler ini nantinya yang berupa *String* akan dikirim ke PC yang sudah di *install* game, yang nantinya permainan ini akan ditampilkan pada layar monitor.

BAB IV

HASIL DAN PEMBAHASAN

Secara umum, pengujian ini bertujuan untuk mengetahui apakah alat yang dapat dibuat dapat bekerja sesuai dengan spesifikasi perencanaan yang telah ditentukan. Pengujian dilakukan untuk mengetahui kinerja alat yang dipakai, karena suatu alat juga pasti akan mengalami penurunan kualitas.

4.1 Hasil Uji Coba

4.1.1 Hasil Kalibrasi

Hasil kalibrasi sensor tentunya akan menggunakan *Arduino Development Environment*, berikut hasil kalibrasi dari tiap sensor.

1. Hasil kalibrasi sensor kemudi dan rem

- Data yang akan diperoleh ialah *value* dari kalibrasi -90 (belok kiri) dan 90 (belok kanan), kalibrasi kemudi dengan tahap setir/setang kemudi di belokkan ke kiri 90° dan kemudian pada kolom inputan serial monitor cukup menekan huruf **A** dan tekan *enter*.

```
if(perintah=='a')//kalibrasi lowIn (-90 derajat)
            {
                    lowIn = analogRead(A0);
                    //masukkan lowIn ke EEPROM;
                    if(EEPROM.updateInt(0, lowIn))
                    {

                            kirim   =   (String)"info#Analaog   value
"+lowIn+" as -90 calibration";
                            Serial.println(kirim);
                    }
            }
```

Potongan kode di atas adalah hasil kalibrasi kemudi

Untuk belok kanan prosesnya sama, hanya saja untuk pada kolom *inputan* ditekan **B** dan *enter*.

```
else if(perintah=='b')//kalibrasi highIn (90 derajat)
            {
                    highIn = analogRead(A0);
                    //masukkan highIn ke EEPROM;
                    if(EEPROM.updateInt(2, highIn))
                    {
                                kirim = (String)"info#Analaog value
"+highIn+" as 90 calibration";
                                Serial.println(kirim);
                    }
            }
```

Potongan kode di atas adalah hasil kalibrasi kemudi

Maka nanti akan diketahui *value* dari data kiri dan kanan. Sedangkan pemberian tombol A (-90) dan kanan (90) ini bertujuan untuk memudahkan proses kalibrasi, dengan memberikan koding pada program yang sudah ter-*upload* pada mikrokontroler.

```
-90.00
info#Analaog value 204 as -90 calibration
-90.00
```

Gambar 4. 1 hasil kalibrasi belok kiri

Dari gambar di atas merupakan hasil kalibrasi dari -90° (belok kiri), sedangkan untuk hasil belok kanan (90°) dapat dilihat pada gambar dibawah.

```
90.00
info#Analaog value 920 as 90 calibration
90.00
```

Gambar 4. 2 hasil kalibrasi belok kanan

- Sedangkan pada kalibrasi rem prosesnya sama dengan kalibrasi kemudi belok kiri (-90), hanya belok kanan (90). Cuman pada keadaan rem tidak ditekan/ rem

tidak digunakan akan maka cukup menakan huruf **E** dan *enter* untuk memunculkan *value*.

```
else if(perintah=='e')//kalibrasi lowRem (tidak ngerem)
            {
                    lowRem = analogRead(A3);
                    //masukkan lowIn ke EEPROM;

                    if(EEPROM.updateInt(6, lowRem))
                    {
                            kirim = (String)"info#Analaog value
"+lowRem+" as lowRem calibration";
                            Serial.println(kirim);
                    }
            }
```

Potongan kode di atas adalah hasil kalibrasi rem *low*

Sedangkan ketika rem pada posisi digunakan maka cukup menekan huruf **F** dan *enter*.

```
else if(perintah=='f')//kalibrasi highRem (sangat ngerem)
            {
                    highRem = analogRead(A3);
                    //masukkan highIn ke EEPROM;
                    if(EEPROM.updateInt(8, highRem))
                    {
                            kirim = (String)"info#Analaog value
"+highRem+" as highRem calibration";
                            Serial.println(kirim);
                    }
            }
```

Potongan kode di atas adalah hasil kalibrasi rem *high*

Dan berikut adalah hasil *value* dari sensor rem *low* ataupun *high.*

```
#0.00
info#Analaog value 13 as lowRem calibration
#0.00
```

Gambar 4. 3 kalibrasi sensor rem dalam keadaan tidak mengerem*/ low*

```
#9.00
info#Analaog value 144 as highRem calibration
#9.00
```

Gambar 4. 4 kalibrasi sensor rem dalam keadaan menekan rem/ *high*

2. Hasil kalibrasi sensor kecepatan

Untuk kalibrasi kecepatan dengan cara posisi sensor berada pas pada warna putih, maka ditekan tombol **C**.

```
else if(perintah=='c')
            {
                  batasAnalogWhite =  analogRead(A1);
                  kirim     =     (String)"info#Analaog     value
"+batasAnalogWhite+" as white calibration";
                  Serial.println(kirim);
            }
```

Potongan kode di atas adalah hasil kalibrasi kecepatan sensor putih

Sedangkan untuk warna hitam ditekan tombol **D** lalu *enter*.

```
else if(perintah=='d')
            {
                  int batasAnalogBlack =  analogRead(A1);
                  kirim     =     (String)"info#Analaog     value
"+batasAnalogBlack+" as black calibration";

                  Serial.println(kirim);
                  batasBlackWhite   =   (int)(batasAnalogBlack   -
batasAnalogWhite)/2; // batas tengah antara black white

                  if(EEPROM.updateInt(4, batasBlackWhite))
                    {
                              kirim = (String)"info#Analaog value
"+batasBlackWhite+" as batasBlackWhite calibration";
                              Serial.println(kirim);
                    }
            }
```

Potongan kode di atas adalah hasil kalibrasi kecepatan sensor hitam

Berikut gambaran hasil *value* kaibrasi kecepatan.

```
#0.00
info#Analaog value 64 as white calibration
#0.00
```

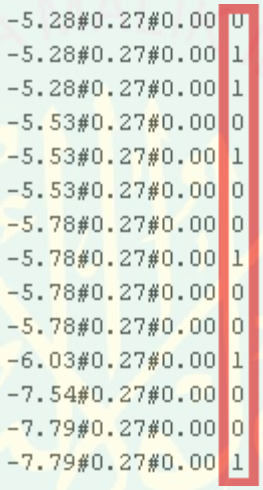
Gambar 4. 5 kalibrasi pada titik putih

```
#0.00
info#Analaog value 794 as black calibration
info#Analaog value 365 as batasBlackWhite calibration
#0.00
```

Gambar 4. 6 kalibrasi pada titik hitam

3. Hasil uji tombol *push on*

Sedangkan pada tombol *Push on* ini, tidak ada koding yang digunakan pada mikrokontroler. Hanya memanggil dari *port* A2 yang sudah berada pada mikrokontroler. Jadi tombol ini langsung tertancap pada *port* yang sudah disediakan pada *Arduino Duemelanove*. berikut adalah hasil uji coba tombol pada *Arduino IDE*.



Gambar 4. 7 hasil uji coba tombol *push on*

Gambar di atas merupakan hasil uji coba ketika tombol *push on* dijalankan, nilai 1 menandakan tombol ditekan dan nilai 0 bahwa tombol tidak ditekan. Semua uji coba kalibrasi ini menggunakan *Arduino IDE*, dan data masuk dari sensor nanti akan bisa dilihat hasil kalibrasi pada *value*, dan *value* inilah yang menjadi data masukan pada koding di *arduino.*

4.1.2 Hasil uji komunikasi serial Asinkron

Data hasil perhitungan di mikrokontroler.

```
kirim=(String)derajatKemudi()+"#"+Rem()+"#"+kecepatanMperSecond+"#"+digi
talRead(A2);
```

Potongan kode di atas adalah kalibrasi kecepatan

Proses pengiriman.

- Misal data hasil dari mikrokontroler: 45#5#6#0 dalam satu blok pengiriman
- Data hasil akan dikirim satu persatu menjadi: $\underset{0}{4}\ \underset{1}{5}\ \underset{2}{\#}\ \underset{3}{5}\ \underset{4}{\#}\ \underset{5}{6}\ \underset{6}{\#}\ \underset{7}{0}$
- Maka data akan dikirim satu per satu dengan i = 0; i < length (kirim);i++. Pengiriman ini berlanjut dalam data satu blok.
- Maka diketahui pengiriman pertama yaitu 4, dan di jadikan 0000 0100 menjadi bit. untuk megubah menjadi data bit ialah protokol di mikrokontroler.
- pada proses inilah tranmisi *asinkron* dijalankan untuk memberikan *startbit*(0)+*stopbit*(1), maka menjadi 0000001001.
- Maka 0000001001 ini menjadi databit yang dikirim pada *tranmisi asinkron*.

Proses penerima.

- Data bit yang diterima dari tranmisi *asinkron* 0000001001 masih data bit.
- Maka data akan diubah menjadi data bit semula, dengan Dbit =terima-*startbit-stopbit*. Data kan kembali manjadi 0000 0100
- Maka data bit 0000 0100 akan di ubah kembali pada protokol menjadi 4 dalam satuan desimal.

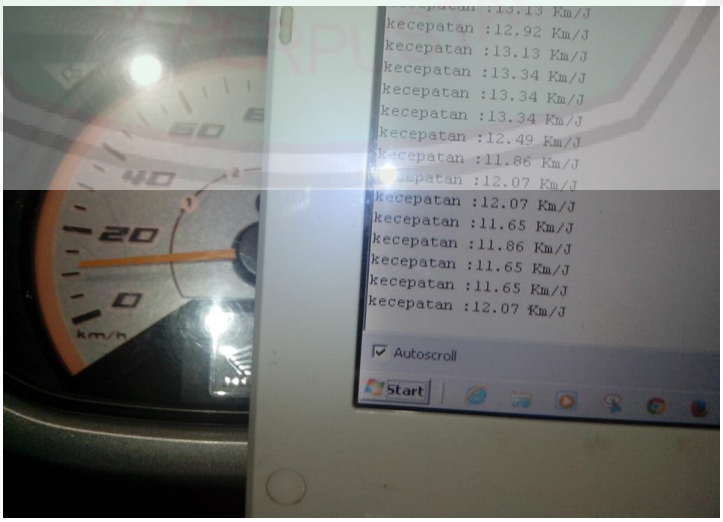
Proses ini akan terus menerus dikirim ketika data dari mikrokontroler masih ada dengan kecepatan pengiriman data 9600.

Dalam proses mengubah bilangan desimal menjadi bit dengan protokol yang sudah tersedia pada mikrokontroler *Arduino Duemelanov* dan juga pada PC/komputer. Sehingga data akir dari pengiriman inilah yang akan di proses pada *game* menjadi inputan.

4.1.3 Hasil uji coba statistik data

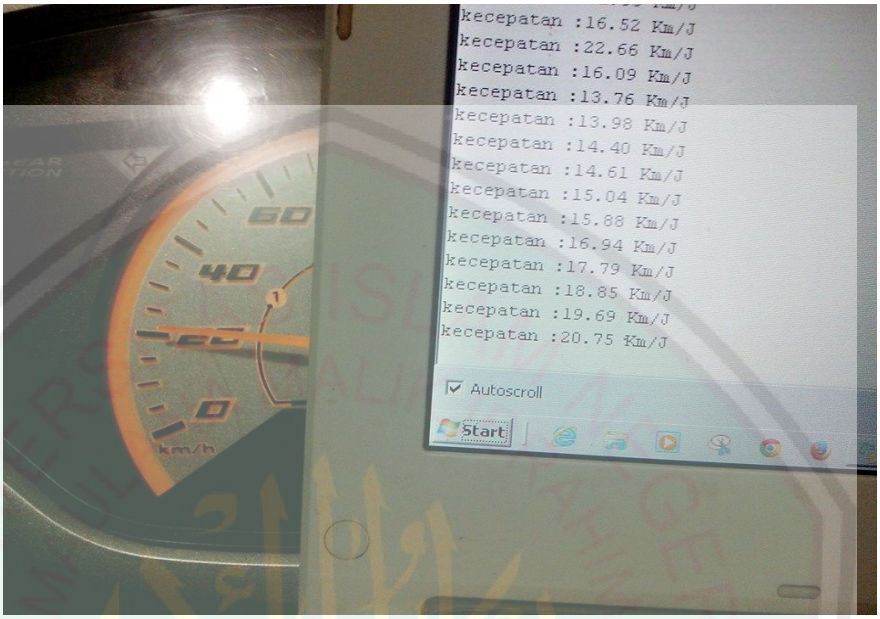
Pada uji coba ini akan membahas tentang hasil data *statistik* dari sensor kecepatan. Dimana memakai sensor kecepatan Infra merah dan *photo dioda* yang di tempelkan pada roda motor, dimana *speedometer* dari motor ini masih standar dan berfungsi normal. Pada koding *arduino* akan dirubah untuk diameter roda dan jumlah tanda hitam dan putih, karena hal tersebut berbeda antara sepeda dengan motor. Dapat diketahui uji coba data *statistik* dari sensor dengan uji coba berikut.

1. Uji coba pertama



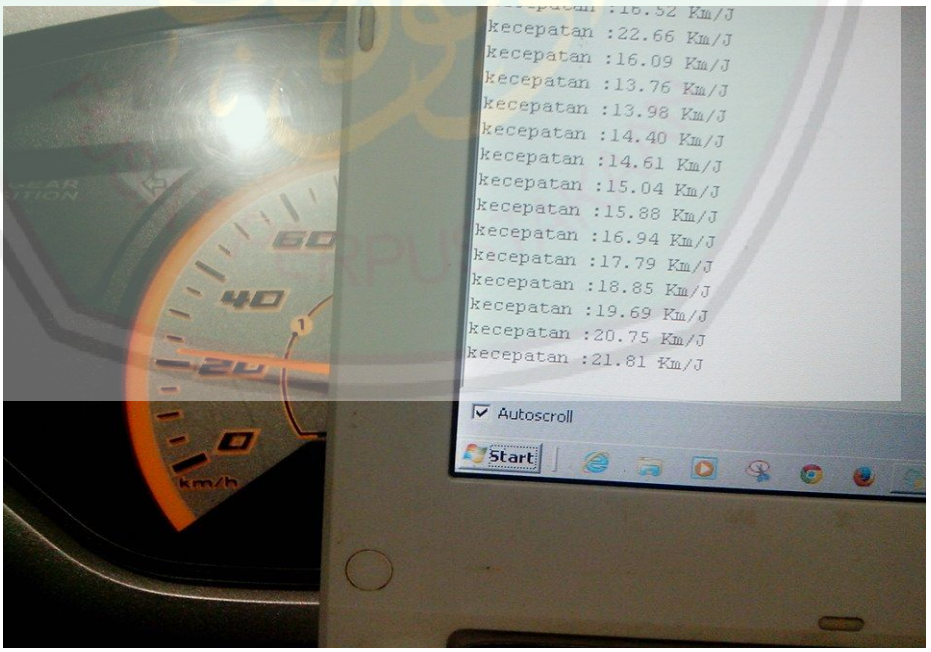
Gambar 4. 8 hasil data *speedometer* dan sensor pada uji coba

2. Uji coba kedua



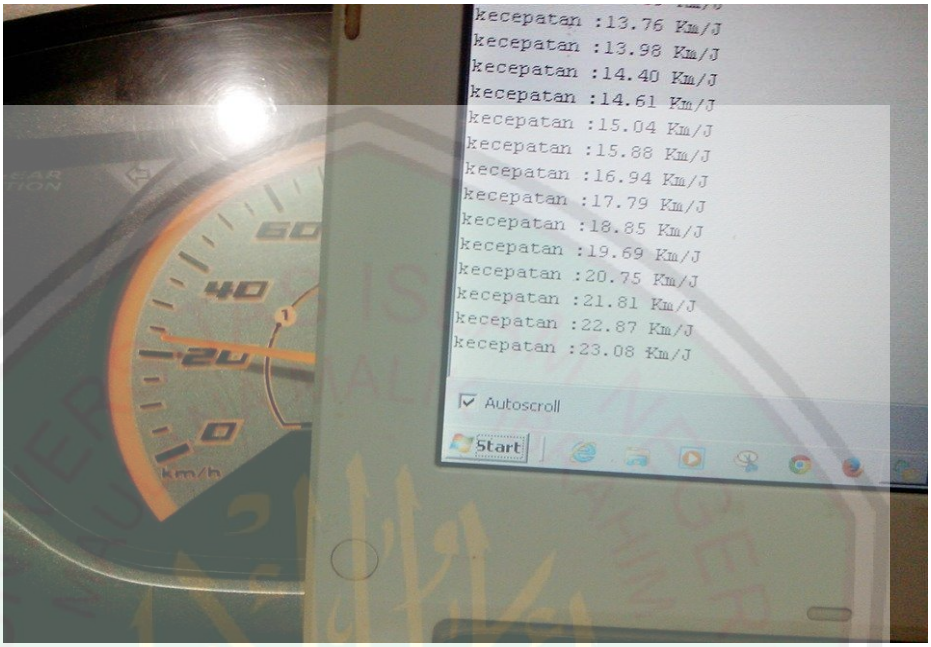
Gambar 4. 9 hasil data *speedometer* dan sensor pada uji coba

3. Uji coba ketiga



Gambar 4. 10 hasil data *speedometer* dan sensor pada uji coba

4. Uji coba keempat



Gambar 4. 11 hasil data *speedometer* dan sensor pada uji coba

Gambar hasil uji di atas merupakan hasil antara data sensor dengan hasil *speedometer*. Dan berikut rekapitulasi hasil antara data sensor dengan *speedometer*.

NO	*Speedometer* km/J	Hasil Sensor (Km/J)
1	12	12.07
2	20	20.75
3	21	21.81
4	23	23.08

Data statistik ini akan diterapkan pada sepeda, yang tinggal mengubah diameter roda dan jumlah tanda hitam maupun putih dan tak lupa untuk mengubah

satuan km/j menjadi rps. Karena kecepatan orang mengayuh tidak sampai melebihi kecepatan sepeda motor, pemakaian sepeda motor ditujukan untuk mencoba hasil data untuk mengurangi selisih. Karena *game* dengan *Immersive Tool* ini diusahakan semirip dengan kenyataan baik untuk jarak tempuh dan juga kecepatan.

4.2 Koneksi Data Multisensor ke *Game*

Koneksi data dari multisensor yang merupakan *inputan* dari *game* diperoleh dari data *digital* dari mikrokontroler yang akan dikirim melalui USB ke PC. Pengiriman Data serial menggunakan USB dari mikrokontroler menuju ke *game* dengan memanggil *port* serial yang terdapat pada koding *game*, berikut koding yang terdapat pada *game* untuk koneksi dengan mikrokontroler.

```
SerialPort stream = new SerialPort ("COM1", 9600); //set com1
portnya dan 9600 baudratenya
            float[] lastRot = {0,0,0};
            Vector4 rot;
            Vector4 offset;

            void Start ()
            {
                        stream.Open (); // membuka koneksi serial
                        //sp.ReadTimeout = 1;
            }

            // Update is called once per frame
            void Update ()
            {
                        amountToMove = speed * Time.deltaTime;
                        string value = stream.ReadLine ();
//membaca data
                        string[] vec4 = value.Split ('#'); //split
data yang di kirimkan oleh arduino menjadi parameter baru
                        if (vec4 [0] != "" && vec4 [1] != "" &&
vec4 [2] != "" && vec4 [3] != "") {
                                    belok = float.Parse (vec4
[0]);
                                    rem = float.Parse (vec4 [1]);
                                    rpmx = float.Parse (vec4 [2]);
```

```
                                    mundur = float.Parse (vec4
[3]);
                                //    Debug.Log ("belok : " + belok
+ "| rpmx : " + rpmx + "m/s | rem : " + rem + " | mundur : " +
mundur);
                                    stream.BaseStream.Flush ();
//Clear the serial information so we assure we get new
information.
                            }
                            RodaBelakngTrans.Rotate (RodaBelakang.rpm
/ 60 * 360 * Time.deltaTime, 0, 0);
                            Steer.localEulerAngles = new Vector4 (0,
RodaDepan.steerAngle, 0, 0);
                            RodaDepanTrans.localEulerAngles = new
Vector4 (0, RodaDepan.steerAngle, 0, 0);
                            RodaDepanTrans.Rotate (0,
RodaDepan.steerAngle, 0);
                            Steer.Rotate (0, RodaDepan.steerAngle, 0);
                Debug.Log ("belok :" + belok + " | rem :" + rem + " |
rpm :" + rpmx);
                }
```

Potongan kode di atas adalah memanggil *port* serial

Bisa kita lihat dalam koding di atas merupakan koding untuk koneksi dan membaca data serial yang dikirim dari mikrokontroler, dan data yang merupakan data digital dari sensor yang sudah melalui ADC dan program yang sudah di upload pada mikrokontroler akan menjadi inputan paga *game*. Dalam pengaturan komputer nantinya dapat dilihat ketika kabel USB yang menghubungkan *Arduino* dan komputer dicolokkan, nanti tinggal kita lihat pada *Device Manager*. Nanti terdapat USB serial Port dan untuk koding memanggil di *game* akan disamakan, misal pada koding di atas terdapat ” *SerialPort* ("COM1", 9600)”.

Sedangkan pada tampilan layar *game* nantinya terdapat *speedometer* yang dijadikan tampilan data statistik dari sensor kecepatan, data *inputan* dari *speedometer* ialah data kecepatan dari sensor kecepatan yang sudah dikalibrasi dan sudah dicari kecepatan per detik dalam koding mikrokontroler. dan berikut koding dalam *game* tentang *speedometer*.

```
var dialTex: Texture2D;
var needleTex: Texture2D;
var dialPos: Vector2;
var topSpeed: float = 80;
var stopAngle: float = -180;
var topSpeedAngle: float = 0;
var RodaBelakang : WheelCollider;
var currentSpeed : float = 50;

//pause menu
var isPause = false;
var arrow = true;
var sizex : float = 200;
var sizey : float = 130;
function Update (){
      if ( Input.GetKeyDown(KeyCode.Escape))
      {
            isPause = !isPause;
            if(isPause)
                  Time.timeScale = 0;
            else
                  Time.timeScale = 1;

      }
}

//spedo
function OnGUI() {
      currentSpeed                                                    =
2*22/7*RodaBelakang.radius*RodaBelakang.rpm*60/1000;
      GUI.DrawTexture(Rect(dialPos.x,dialPos.y,     dialTex.width,
dialTex.height), dialTex);
      var   centre   =   Vector2(dialPos.x   +   dialTex.width   /   2,
dialPos.y + dialTex.height / 2);
      var savedMatrix = GUI.matrix;
      var speedFraction = currentSpeed / topSpeed;
      var   needleAngle   =   Mathf.Lerp(stopAngle,   topSpeedAngle,
speedFraction);
      GUIUtility.RotateAroundPivot(needleAngle, centre);
      GUI.DrawTexture(Rect(centre.x, centre.y - needleTex.height /
2, needleTex.width, needleTex.height), needleTex);
      GUI.matrix = savedMatrix;
//pause menu
if (isPause) {
      GUI.Window(0,new        Rect        (Screen.width/2-sizex/2,
Screen.height/2-sizey/2,   sizex,   sizey),   TheMainMenu,   "Pause
Menu");

      }
}

function TheMainMenu (){
      if(GUILayout.Button("MainMenu")){
      Application.LoadLevel(1);
}
if(arrow){
```

```
        if(GUILayout.Button("Enable Navigation")){
        Time.timeScale = 1;
        isPause = !isPause;
        arrow = !arrow;
        GameObject.Find("arrow").SetActive(true);
        }
}
else{
        if(GUILayout.Button("Disable Navigation")){
        Time.timeScale = 1;
        isPause = !isPause;
        arrow = !arrow;
        GameObject.Find("arrow").SetActive(false);
        }
}
if(GUILayout.Button("Restart")){
        Application.LoadLevel(2);
}

if(GUILayout.Button("Quit")){
Application.Quit();
}
}
```

Potongan kode di atas adalah *speedometer* pada *game*

Data inputan dari *speedometer* pada *game* ini berupa data hasil yang dikirimkan dari mikrokontroler yang berupa kecepatan tiap detik (rps). Dalam data statistik kecepatan yang ditampilkan melalui *speedometer* ini diharapkan bisa melihat data kecepatan ketika mengayuh.

4.3 Pembahasan

Setelah didapatkan hasil pengujian kemudian hasil pengujian tersebut dibahas agar didapatkan pengetahuan dan ilmu baru berdasarkan hasil penelitian.

Berikut ini adalah data hasil dari penelitian ini.

1. Pembahasan tentang hasil kalibrasi saat dianjurkan untuk awal mula suatu *game* yang menggunakan *Immersive Tool*, hal ini ditujukan untuk menentukan kebenaran nilai konvensional penunjuk suatu instrumen alat ukur. Jadi kalibrasi dilakukan sebelum bermain, untuk meriset awal mula data yang dipakai.
2. Data dari sensor yang sudah dikalibrasi pada mikrokontroler akan dikirim ke PC dan merupakan inputan dari *game* sepeda, dalam pengiriman data serial ini menggunakan USB dan komunikasi serial *Asinkron*.
3. Data statistik sensor yang berupa hasil dari *speedometer* nantinya akan berupa tampilan *speedometer* yang terdapat pada tampilan *game*. *Speedometer* ini bertujuan untuk menunjukkan data kecepatan rata-rata dan maksimal *player* dalam mengayuh sepeda.

4.4 Integrasi *Game* sepeda menggunakan *Immersive Tool* dengan Islam

Suatu *game* pada dasarnya sebagai sarana dan prasarana hiburan, baik melepas lelah ketika habis kerja atau semua aktifitas lainya. Berikut ini integrasi *game* sepeda menggunaakan *Immersive Tool* dengan Islam.

Islam merupakan agama yang sempurna segala lini kehidupan diatur olehnya, bahkan tentang berolahraga pun ada dijelaskan. Anjuran ini tidak lain agar manusia memilki tubuh yang kuat dan sehat, sehingga dapat optimal beribadah kepada Allah. Dalil yang menjelaskan tentang olahraga antara lain:

وَأَعِدُّوْا لَهُمْ مَااسْتَطَعْتُمْ مِنْ قُوَّةٍ وَمِنْ رِبَاطِ الْخَيْلِ تُرْهِبُوْنَ بِهِ عَدُوَّ اللهِ وَعَدُوَّكُمِ وَآخَرِيْنَ مِنْ دُوْنِهِمْ.

"Dan persiapkanlah dengan segala kemampuan untuk menghadapi mereka dengan kekuatan yang kamu miliki dan dari pasukan berkuda yang dapat menggentarkan musuh Allah, musuhmu dan orang-orang selain mereka". (QS. *Al-Anfal*/8; 60). (Saputera).

Larangan tegas dari Nabi *shallallahu 'alaihi wa sallam* hanya untuk main dadu. Karena di zaman beliau, permainan itu yang baru dikenal. Melalui sabdanya, Nabi *shallallahu 'alaihi wa sallam* menegaskan:

وَدَمه خِنْزِير لَحْم فِي يَده صَبَغَ فَكَأَنَّمَا بِالنَّرْدَشِيرِ لَعِبَ مَنْ

"Siapa yang bermain dadu, dia seperti mencelupkan tangannya ke daging babi dan darahnya." (HR. Muslim 2260) (Baits).

BAB V

PENUTUP

5.1 Kesimpulan

Kesimpulan yang diperoleh dari penelitian tuagas akir **Design *Immersive Tool* Multisensor pada *Game* Sepeda Berbasis Statistika dan Komunikasi Data Asinkron** adalah:

1. Selain sebagai fungsi utama *game*, *game* yang menggunakan *Immersive Tool* ini juga bisa sarana dan prasarana edukasi dan pembelajaran, juga bisa untuk sarana berolah raga. Suatu *game* bisa lebih memamfaat ketika *game* itu bertujuan baik bagi *player*, karena sekarang banyak *game* yang kurang mendidik.
2. Dalam *game* sepeda yang menggunakan *Immersive Tool* ini juga memberikan pengalaman tersendiri pagi *player*, karena proses bermainnya pun seperti bermain sepeda secara umum yaitu dengan mengayuh, mengerem dan juga berbelok. Jika *player* ingin menambah kecepatan maka cukup menambahkan ayuhan agar kecepatan roda bisa bertambah dan jika ingin berhenti atau ingin berbelok maka cukup membelokkan kemudi dan memakai rem jika perlu.
3. Bermain *game* dengan *Immersive Tool* sangat mengasyikkan dari pada bermaian tanpa *Immersive Tool*, karena akan memberikan kesan yang berbeda bagi *player*. Hal ini memberikan pengalaman yang berbeda bagi

player yang ingin bermain permaian sepeda dengan kendali sepeda sesungguhnya.

4. Pengiriman data serial menggunakan USB pada mikrokontroler ke PC dirasa sangat pas, juga bisa sebagai *power suply* daya di mikrokontroler.

5.2 Saran

Dalam penelitian ini masih terdapat beberapa kekurangan. Untuk pengembangan lebih lanjut terdapat saran-saran sebagai berikut ini.

1. *Immersive Tool* dapat diberi tambahan belok dengan kemiringan, sehingga *player* selain belok dengan kemudi juga bisa degan kemiringan sepeda.
2. Dalam satu *game* bisa diberikan dua pemain atau lebih, jadi bisa *multiplayer*. Sehingga *player* bisa merasakan seakan-akan bersepeda bersama atau juga bisa mengadu kecepatan.
3. Penelitan selanjutnya dapat merubah tampilan sepeda, Berusaha untuk meminilmalkan dan lebih praktis dalam penempatan sensor maupun model *Immersive Tool* sendiri, tetapi tidak mengurangi fungsi dan kegunaan.
4. Supaya memakai sensor dengan kualitas bagus, karena fungsi sensor dalam *game Immersive Tool* sangat fital. Hal ini bertujuan untuk menjaga akselerasi sensor, sehingga mengurangi memudarnya keakuratan suatu sensor.

DAFTAR PUSTAKA

Al – Qur'an Digital versi 2.1 (http://www.alquran-digital.com)

Artanto, Dian. 2012. 60 *Aplikasi menggunakan PLC-Mikro menggunakan* Mikrokontroler *PIC16f877 & Duemilanove (arduino severino)*Elex Media, Mujahidin. 2006. *Pemrograman Port Serial*

Ashaulo, p. (2012). Development of an Interactive 3-D Virtual Environment Tet Bed For Motorbikes. Electrical and Information Technology Research Unit, Savonia University of Applied Sciences .

Baits, A. N. (t.thn.). Dipetik Februari 12, 2014, dari Konsultasi Syariah: http://www.konsultasisyariah.com/hukum-main-catur-haram/ diakses pada 12-04-2014

Hardy, Sandro. *Adaptation Model for Indoor Exergames Technische* Universitas Darmstadt

Heryanto, Ari, dkk.2013. *Pemrograman Bahasa C untuk Mikrokontroler Duemilanove535* Andi Publisher

kong, w. (2010, desember 3). UART – Universal Asynchronous Receiver and Transmitter.

Laros, Edu. 2007. *Komunikasi Serial Antara PC-Mikrokontroler dengan RS232 Komunikasi Serial Antara PC-Mikrokontroler dengan RS232* (5): 3-7

Saputera, M. (t.thn.). Dokter Muslim el-Banjary. Dipetik April 12, 2014, dari http://doktermuslimyonirazer.blogspot.com/2012/10/kesehatan-dalam-konsep-al-quran.html diakses pada 12-04-2014

Scherfgen. *An Immersive Bicycle Simulator* Bonn-Rhine-Sieg University of Applied Sciences, University of New Brunswick, Faculty of Computer Science

Teguh, W. (2003). Prinsip dasar dan Teknologi KOMUNIKASI DATA. yogyakarta.