



**ANALISA *PACKET LOSS* SISTEM TELEMETRI PADA
PERANGKAT PENGUKUR KECEPATAN ANGIN BERBASIS
X-BEE PRO MENGGUNAKAN *KALMAN FILTER***

SKRIPSI

oleh

**Desti Husumardiana
NIM 111910201009**

**PROGRAM STUDI STRATA 1 TEKNIK ELEKTRO
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS JEMBER
2015**

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan zaman dan teknologi kebutuhan informasi yang cepat sangat dibutuhkan dalam berbagai bidang, baik pada bidang pertanian, perikanan, industri, maupun stasiun meteorologi sehingga bisa menunjang kinerja bidang tersebut. Salah satunya adalah informasi tentang parameter pengukuran cuaca seperti suhu, kecepatan angin, dan curah hujan. Namun dalam pemantauan dan pengukuran tidak semua kondisi memungkinkan dilakukan secara langsung dikarenakan faktor geografis dan jarak, hal itu dapat menghambat memperoleh informasi tersebut. Kendala pengukuran pada lokasi yang sulit terjangkau dapat diatasi dengan menggunakan metode pengukuran jarak jauh (telemetry).

Sistem telemetry sering digunakan untuk pengukuran di daerah-daerah yang sukar untuk dijangkau manusia seperti laut, gunung, gua atau lembah. Sistem pengiriman data dengan menggunakan nirkabel dengan penekanan biaya sekecil mungkin menjadi solusi terbaik permasalahan ini. Hal ini dikarenakan sistem telemetry akan mempermudah proses metode pengukuran langsung yang mengharuskan alat pengukur diletakkan pada lokasi yang sulit ditinjau langsung setiap saat, sehingga pengamat tidak perlu pergi ke lokasi untuk mengambil data. Pemantauan yang terus-menerus tidak memungkinkan petugas untuk melakukan pengukuran secara terus-menerus, sehingga petugas cukup meletakkan alat ukur pada tempat pengukuran dan dapat dipantau dari tempat lain. Adanya modul-modul sederhana dengan sistem telemetry (*transmitter-receiver*) dapat dimanfaatkan sebagai pendukung solusi di atas. *Wireless* adalah salah satu teknik komunikasi untuk menyampaikan informasi dengan menggunakan gelombang radio untuk menggantikan kabel yang menghubungkan komputer dengan jaringan, sehingga komputer dapat berkomunikasi dengan jaringan lebih efektif dan efisien serta

dengan kecepatan yang memadai. Kelebihan - kelebihan inilah yang sangat mendukung pemanfaatan *wireless* sebagai media yang digunakan untuk mengakses informasi secara *real time*.

Penelitian sebelumnya yang telah dilakukan oleh Bhakti Dharmawan (2012) dengan judul Analisa Potensi Tenaga Angin Dengan Metode *Weibull Analysis* di Pantai Puger Kabupaten Jember. Masih terdapat kekurangan yaitu pengiriman data dilakukan secara manual. Kekurangan - kekurangan dan latar belakang di atas memberikan ide kepada penulis untuk membuat alat pengukuran kecepatan angin menggunakan sistem telemetri. Namun kita ketahui bahwa sistem yang berbasis *wireless* pasti rentan sekali terhadap faktor lain seperti adanya *noise* juga mempengaruhi kualitas pengiriman data, oleh karena itu penulis menggunakan *Kalman Filter* untuk meminimalisir bahkan menghilangkan adanya gangguan *noise* terhadap data yang akan dikirim. Perancangan ini juga dilengkapi dengan modul RF sebagai media untuk *transfer* data juga dilengkapi perekam data (*data logger*). *Data logger* merupakan sistem yang berfungsi untuk merekam data ke dalam media penyimpanan data, *data logger* memiliki kapasitas penyimpanan yang cukup besar. Sistem *data logger* ini dibangun dari modul Arduino sebagai pengendalinya. Dengan adanya *data logger* ini, memudahkan kita untuk melakukan pengambilan data.

1.2 Rumusan Masalah

Dalam penelitian ini ada beberapa hal yang menjadi rumusan masalah diantaranya:

1. Bagaimana merancang *Kalman Filter* untuk mengetahui *packet loss* pada perangkat pengukur kecepatan angin berbasis X-Bee Pro?
2. Bagaimana analisa *packet loss* sistem telemetri pada perangkat pengukur kecepatan angin berbasis X-Bee Pro menggunakan *Kalman Filter*?

1.3 Tujuan

Adapun beberapa tujuan dari penelitian ini yaitu :

1. Merancang *Kalman Filter* untuk mengetahui *packet loss* pada perangkat pengukur kecepatan angin berbasis X-Bee Pro.
2. Menganalisa *packet loss* sistem telemetri pada perangkat pengukur kecepatan angin berbasis X-Bee Pro menggunakan *Kalman Filter*.

1.4 Manfaat

Adapun manfaat yang diperoleh dari penulisan skripsi ini adalah sebagai berikut:

1. Meminimalisir adanya gangguan *noise* pada sistem transmisi data menggunakan metode *Kalman Filter* sehingga mendapatkan data yang lebih akurat.
2. Pemahaman tentang metode *Kalman Filter* sebagai pemroses sinyal yang lebih akurat.

1.5 Batasan Masalah

Untuk lebih memfokuskan permasalahan yang akan dianalisa dalam penelitian skripsi ini, maka akan dibatasi permasalahan-permasalahan yang akan dibahas sebagai berikut :

1. Sensor yang digunakan adalah sensor *Optocoupler*.
2. Modul RF yang digunakan adalah X-Bee Pro.
3. Tidak membahas sisi elektronika.
4. Jarak yang dapat dijangkau oleh modul RF 100 meter.
5. *Filter* digital yang digunakan adalah *Kalman filter*.
6. *Monitoring* atau *output* data ditampilkan ke komputer menggunakan *software* Visual Basic.
7. Pengambilan data dilakukan dalam dua kondisi yaitu *loss space* dan *obstacle*.

1.6 Sistematika Penulisan

Secara garis besar penyusunan proposal skripsi ini adalah sebagai berikut:

BAB 1. PENDAHULUAN

Berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan pembahasan, manfaat pembahasan dan sistematika pembahasan.

BAB 2. TINJAUAN PUSTAKA

Berisi tentang tinjauan pustaka yang menguraikan pendapat-pendapat atau hasil-hasil penelitian terdahulu yang berkaitan dengan penelitian yang dilakukan, landasan teori merupakan penjabaran dari tinjauan pustaka.

BAB 3. METODOLOGI PENELITIAN

Menjelaskan tentang metode kajian yang digunakan untuk menyelesaikan skripsi.

BAB 4. HASIL DAN PEMBAHASAN

Berisi hasil penelitian dan analisa hasil penelitian.

BAB 5. PENUTUP

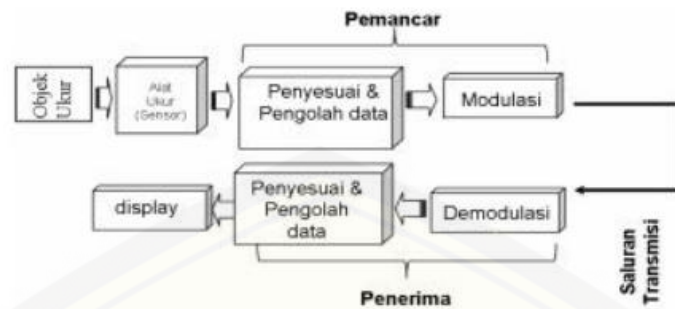
Berisi tentang kesimpulan yang diperoleh dari pembuatan skripsi ini dan saran dari penulis untuk pengembangan lebih lanjut.

BAB 2. TINJAUAN PUSTAKA

2.1 Sistem Telemetry

Telemetry adalah sebuah teknologi pengukuran parameter suatu obyek (benda, ruang, dan kondisi alam) yang hasil pengukurannya di kirimkan ke tempat lain melalui proses pengiriman data baik dengan menggunakan kabel maupun tanpa menggunakan kabel (*wireless*) dilakukan dari jarak jauh dan melaporkan informasi kepada perancang atau *operator system*, selanjutnya data tersebut untuk dimanfaatkan langsung atau perlu dianalisa. Kata telemetry berasal dari bahasa Yunani yaitu *tele* artinya jarak jauh sedangkan *metron* artinya pengukuran. Secara istilah telemetry diartikan sebagai suatu bidang keteknikan yang memanfaatkan instrumen untuk mengukur panas, radiasi, kecepatan atau properti lainnya dan mengirimkan data hasil pengukuran ke penerima yang letaknya jauh secara fisik, berada diluar dari jangkauan pengamat atau *user*. Secara umum sistem telemetry terdiri atas enam bagian pendukung yaitu obyek ukur, sensor, pemancar, saluran transmisi, penerima dan tampilan atau *display*.

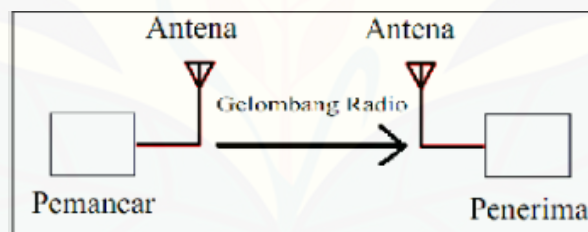
Telemetry dalam keadaan bergerak berpengaruh pada saat pengukuran, pengukuran tersebut untuk mendapatkan nilai percepatan pada suatu benda bergerak. Telemetry bergerak sangat rentan terhadap *noise*. *Noise* yang sering terjadi adalah *noise* dari getaran, suhu, tekanan atmosfer, dan benda yang menjadi penghalang.



Gambar 2.1 Sistem Telemetri
(Sumber : Yudi Triawan, 2011)

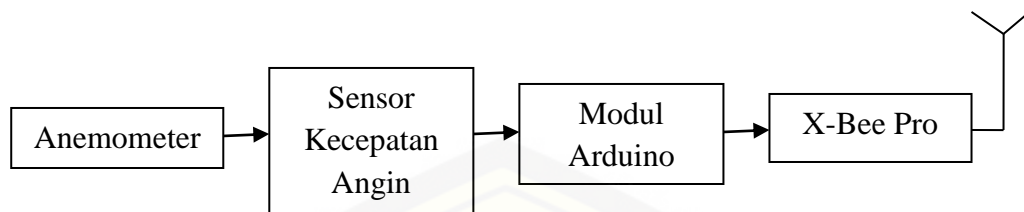
2.1.1 Sistem Pemancaran (*Transmit*)

Sistem pemancar merupakan hal yang pasti ada di suatu sistem telemetri. Baik berupa sistem sederhana maupun sistem yang lebih rumit. Sistem ini sebagai bentuk awal berlangsungnya sistem telemetri, karena informasi dari suatu titik dikirimkan berupa gelombang elektromagnetik pada frekuensi tertentu menggunakan sistem pemancar.



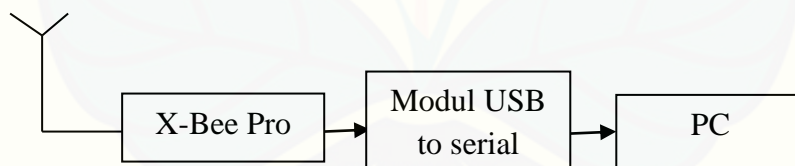
Gambar 2.2 Sistem Pemancaran
(Sumber : Yudi Triawan, 2011)

Ada dua komponen utama dalam sistem ini, yaitu pemancar itu sendiri dan antenna disamping unsur pemasangan dan penyelarasan alat. Antena merupakan bagian dari sistem ini yang berfungsi sebagai radiator gelombang radio. (Yudi Triawan, 2011)

Gambar 2.3 Blok Diagram *Transmitter*

2.1.2 Sistem Penerima (*Receiver*)

Sistem penerima mendapat data dari sistem pemancar. Data yang diterima sebisa mungkin bebas dari faktor eror yang mungkin terjadi pada saat data dikirim dari pemancar maupun pada saat diterima oleh sistem penerima (RF System Committee 2008). Sama dengan sistem pemancar, ada dua komponen utama yang ada pada sistem pemancar, yaitu antenna dan alat penerima itu sendiri. Disini antenna berguna sebagai penerima gelombang radio sedangkan pada alat penerima memiliki berbagai elemen fungsi. Diantaranya adalah penyetel frekuensi yang dikombinasikan langsung dengan antenna, penyaring data, penerjemah data, dan penghitung data sehingga mudah diolah menjadi sebuah produk informasi pada saat masuk ke sistem komputasi.

Gambar 2.4 Blok Diagram *Receiver*

2.2 Alat Pengukur Kecepatan Angin

Anemometer adalah alat uji angin yang digunakan untuk mengukur kecepatan angin, dan merupakan instrumen stasiun cuaca umum. Istilah ini berasal dari kata Yunani “*anemos*”, yang berarti angin. *Anemos* digunakan untuk menggambarkan setiap *instrumen* pengukuran kecepatan udara yang digunakan dalam *meteorologi* atau *aerodinamis*.

Prinsip kerja anemometer sebagai alat ukur kecepatan angin ini sebenarnya sederhana yaitu pada saat alat tertiuip oleh angin, maka mangkok atau baling-baling pada alat tersebut akan bergerak sesuai dengan arah angin. Kecepatan baling-baling atau mangkok tersebut tentu akan semakin cepat jika kecepatan angin semakin besar. Kecepatan angin tersebut dapat diketahui dari jumlah putaran baling-baling setiap detik. Karena dampak dari pengukuran kecepatan angin ini sangatlah besar, maka setiap pengukurannya harus dijamin keakuratannya.

Menurut Sathyajith Mathew dalam bukunya yang berjudul *Wind Energy* menyebutkan bahwa “*The indicators discussed above, along with available wind data from meteorological stations, can give us an idea on the suitability of a given site for wind energy extraction*”. Anemometer dipasang pada tiang-tiang tinggi yang digunakan untuk pengukuran angin tersebut. Ketinggian tiang memungkinkan posisi turbin untuk menghindari koreksi lebih lanjut dengan permukaan. Sebagai kekuatan yang sensitif terhadap kecepatan angin, anemometer berkualitas baik adalah anemometer yang sensitif, dapat diandalkan dan dikalibrasi dengan benar dan harus digunakan untuk pengukuran angin. Dalam penelitian ini yang akan kita gunakan adalah anemometer mangkok (*Cup anemometers*).

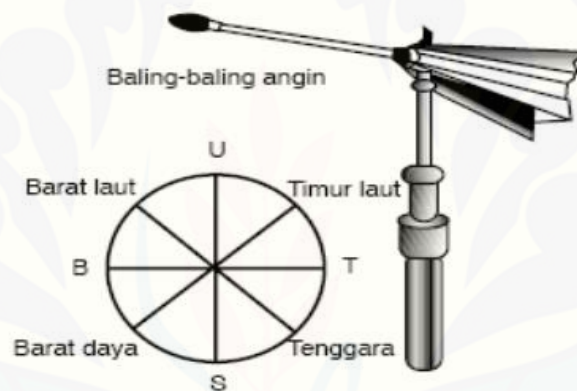
Cup Counter Anemometer adalah alat untuk mengukur kecepatan angin. Angin adalah pergerakan udara pada horizontal atau hampir horizontal. Angin mempunyai arah (*direction*) dan kecepatan (*speed*). Arah angin dinyatakan dari datangnya misalnya : Angin barat (angin yang datang dari barat) dan angin tenggara (angin yang datang dari tenggara. Arah angin (Derajat ukur) Utara = 0/360, Timur = 900, Selatan = 1800, Barat = 2700, Arah angin dinyatakan dalam satuan derajat dan kecepatan angin dinyatakan dalam m/s, km/jam, mil/jam, knots hubungan antara masing-masing satuan ini adalah :

- a. $1 \text{ m/s} = 3.6 \text{ km/jam} = 2 \text{ knots}$
- b. $1 \text{ km/jam} = 10/36 \text{ m/s} = 0.62 \text{ mil/jam}$

c. 1 mil/jam = 0.447 m/s = 1.6 km/jam

d. 1 knots = 0.5 m/s = 1.8 km/jam

Agar dapat membandingkan pengamatan angin yang dilakukan di berbagai tempat, maka pemasangan anemometer dan *vane* tidak boleh sembarangan. Alat ini di pasang pada ketinggian yang sama di atas tanah terbuka. Tanah terbuka adalah lapangan dengan benda (Pohon, rumah, dll) yang berjarak 10 kali lebih tinggi benda itu dari tiang anemometer. Tinggi yang telah di setuju adalah 10 meter. Arah angin diukur dengan *wind vane*. Kecepatan angin diukur dengan *wind speed* anemometer. (Dharmawan , 2014)



Gambar 2.5 Pengukuran dan Arah Anemometer

(Sumber : Alat Pengukur Kecepatan Angin (Dharmawan , 2014))

2.3 Data Logger

Data logger menyimpan data teknis dan sensor. Sebuah data *logger* (juga *data logger* atau *data recorder*) adalah perangkat elektronik yang mencatat data dari waktu ke waktu atau dalam kaitannya dengan lokasi baik dengan *built in instrumen* atau sensor atau melalui instrumen eksternal dan sensor. Namun tidak sepenuhnya *data logger* didasarkan pada prosesor digital (atau komputer). *Data logger* umumnya adalah kecil, bertenaga baterai, portabel, dan dilengkapi dengan mikroprosesor, memori internal untuk penyimpanan data, dan sensor. Beberapa *data logger* antarmuka dengan komputer pribadi dan memanfaatkan perangkat

lunak untuk mengaktifkan *data logger* dan melihat serta menganalisis data yang dikumpulkan, sementara yang lain memiliki perangkat antarmuka lokal (*keypad*, LCD) dan dapat digunakan sebagai perangkat yang berdiri sendiri.

Data logger bervariasi antara jenis tujuan umum untuk berbagai aplikasi pengukuran untuk perangkat yang sangat spesifik untuk mengukur dalam satu lingkungan atau jenis aplikasi saja. Hal ini umum untuk jenis tujuan umum harus diprogram, namun, masih banyak sebagai mesin statis dengan hanya sejumlah terbatas atau tidak ada parameter berubah. *Data logger* elektronik telah menggantikan perekam grafik dalam banyak aplikasi. Salah satu manfaat utama menggunakan *data logger* adalah kemampuan untuk secara otomatis mengumpulkan data pada basis 24 jam. Setelah aktivasi, *data logger* biasanya digunakan dan ditinggalkan untuk mengukur dan merekam informasi selama periode pemantauan. Hal ini memungkinkan untuk gambar, komprehensif akurat dari kondisi lingkungan yang dipantau, seperti kecepatan angin. (Pribadi, 2011)

2.4 Modul Arduino Uno Atmega 328P

Arduino Uno Atmega 328P memiliki 14 digital *input/output* pin (dimana 6 diantaranya dapat digunakan sebagai *output* PWM), 6 *input* analog, osilator 16 MHz kristal, koneksi USB, soket listrik, *header* ICSP, dan tombol *reset*. Ini berisi semua yang diperlukan untuk mendukung mikrokontroler, hanya dengan menghubungkan ke komputer dengan kabel USB atau menghidupkannya dengan adaptor AC-DC atau baterai untuk memulainya. Berikut adalah gambar skema dan indeks *Arduino board* :

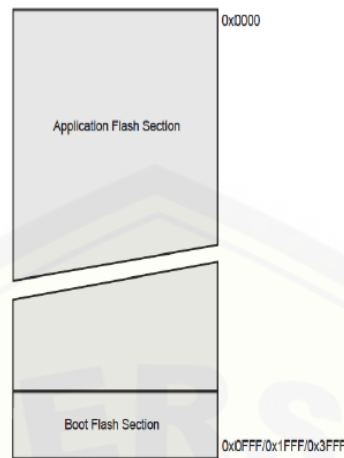


Gambar 2.6 Skema Arduino Uno
(Sumber : www.arduino.cc)

1. *Operating voltage* 5V.
2. Rekomendasi *input voltage* 7-12V.
3. Batas *input voltage* 6-20V.
4. Memiliki 14 buah *digital input/output*.
5. Memiliki 6 buah *Analog Input*.
6. DC *Current* setiap I/O pin sebesar 40 mA.
7. DC *Current* untuk 3,3V *pin* sebesar 50 mA.
8. *Flash Memory* 32 KB.
9. SRAM 2 KB.
10. EEPROM 1 KB.
11. *Clock Speed* 16 MHz. (sumber : www.arduino.cc)

2.4.1 Atmega 328P

Manajemen memori dalam mikrokontroler penting dilakukan karena memori yang dimiliki mikrokontroler sangat terbatas. Pada Atmega328P terdapat tiga jenis memori, yaitu *data memory*, *program memory*, dan EEPROM. Ketiga memori tersebut terpisah, sehingga dapat mengakses ketiga jenis memori tersebut dalam waktu yang bersamaan. Atmega328P menggunakan *Flash Memory* untuk *program memory*. *Flash Memory* dibagi menjadi dua bagian, yaitu *Boot loader* dan *Application Program*. Pembagian ini bertujuan untuk keamanan perangkat lunak. *Flash Memory* memiliki ketahanan tulis atau hapus sebanyak 10.000 kali. SRAM digunakan oleh Atmega328P untuk *data memory*.



Gambar 2.7 Program *memory map* ATmega328P

(Sumber : Afdhol Arriska, 2012)

Kapasitas SRAM dari Atmega328P adalah 2KB. SRAM terbagi menjadi empat bagian yaitu 32 GPR (*General Purpose Register*), 64 I/O register, *Additional I/O register*, dan *Internal SRAM*. Sifat dari memori ini adalah *volatile* sehingga data yang ada pada SRAM akan hilang ketika sudah tidak diberikan catu daya.

2.4.2 ADC (*Analog to Digital Converter*)

ADC atau kepanjangan dari *Analog to Digital Converter* merupakan alat yang digunakan untuk mengubah data analog menjadi data digital. Pada Arduino Uno sudah terdapat modul ADC, sehingga dapat langsung digunakan. Fitur ADC yang terdapat pada ATmega328P adalah sebagai berikut:

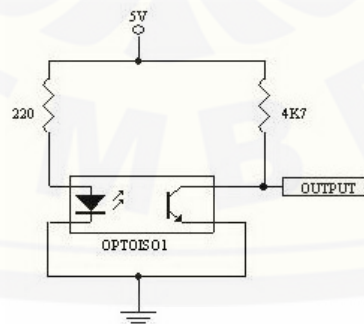
1. Resolusi mencapai 10 bit.
2. 0,5 *LSB Integral Non-linearity*.
3. Akurasi mencapai ± 2 *LSB*.
4. Waktu konvensi 13-260 μ s.
5. Mempunyai 6 saluran ADC.
6. *Optional Left Adjustment* untuk pembacaan hasil ADC.
7. 0 – Vcc untuk kisaran *input* ADC.

8. Disediakan 1,1 V tegangan referensi ADC.
9. Mode konversi kontinyu atau konversi.
10. Interupsi ADC.
11. *Sleep mode noise canceler*

Sinyal *input* dari *port* ADC akan dipilih oleh *multiplexer* (*register* ADMUX) untuk diproses oleh ADC. Karena *converter* ADC dalam *chip* hanya satu buah sedangkan saluran masukannya lebih dari satu, maka dibutuhkan *multiplexer* untuk memilih *input port* ADC secara bergantian. ADC mempunyai rangkaian untuk mengambil sampel dan *hold* (menahan) tegangan *input* ADC, sehingga tetap dalam keadaan konstan selama proses konversi. Sinyal *input* ADC tidak boleh melebihi tegangan referensi. (Afdhol Arriska, 2012).

2.5 Sensor *Optocoupler*

Optocoupler adalah suatu piranti yang terdiri dari 2 bagian yaitu *transmitter* dan *receiver*, yaitu antara bagian cahaya dengan bagian deteksi sumber cahaya terpisah. Biasanya *optocoupler* digunakan sebagai saklar elektrik, yang bekerja secara otomatis. *Optocoupler* atau optoisolator merupakan komponen penggandeng (*coupling*) antara rangkaian *input* dengan rangkaian *output* yang menggunakan media cahaya (*opto*) sebagai penghubung. Dengan kata lain, tidak ada bagian yg konduktif antara kedua rangkaian tersebut. *Optocoupler* sendiri terdiri dari 2 bagian, yaitu *transmitter* (pengirim) dan *receiver* (penerima).

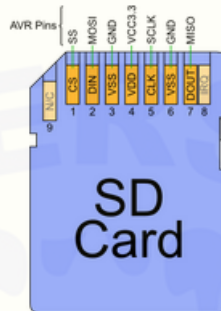


Gambar 2.8 Sensor *Optocoupler*

(Sumber : <http://elektronika-dasar.web.id/>)

2.6 SD Card

MicroSD adalah kartu memori *non-volatile* yang dikembangkan oleh SD Card Association yang digunakan dalam perangkat *portable*. Saat ini, teknologi microSD sudah digunakan oleh lebih dari 400 merek produk serta dianggap sebagai standar industri *defacto*.



Gambar 2.9 SD card

(Sumber : <http://icomputer.com/>)

Keluarga microSD yang lain terbagi menjadi SDSC yang kapasitas maksimum resminya sekitar 2GB, meskipun beberapa ada yang sampai 4GB. SDHC (*High Capacity*) memiliki kapasitas dari 4GB sampai 32GB. Dan SDXC (*Extended Capacity*) kapasitasnya di atas 32GB hingga maksimum 2TB. Keberagaman kapasitas seringkali membuat kebingungan karena masing-masing protokol komunikasi sedikit berbeda.

Dari sudut pandang perangkat, semua kartu ini termasuk kedalam keluarga SD. SD adapter memungkinkan konversi fisik kartu SD yang lebih kecil untuk bekerja di slot fisik yang lebih besar dan pada dasarnya ini adalah alat pasif yang menghubungkan pin dari microSD yang kecil ke pin adaptor microSD yang lebih besar. (Dharmawan, 2014)

2.7 Komunikasi Data Serial

Perkembangan komunikasi di masa sekarang sangat cepat. Dimulai dari teknik komunikasi data secara paralel sampai pengembangan teknik komunikasi

data serial yang dilakukan pengembangan sangat cepat. Pada awalnya yang kita kenal port paralel (DB25) sebagai piranti komunikasi komputer dengan printer dan berbagai alat lainnya. Akan tetapi dilihat dari perkembangan kedepannya, teknik ini mengalami kesulitan dan kendala, hal ini dipandang dari sudut ekonomisnya. Dengan jumlah jalur komunikasi yang banyak, menjadikan komunikasi paralel ini mulai ditinggalkan dan beralih ke teknik komunikasi serial.

Komunikasi data serial dimulai dari port serial (DB9) sampai saat ini dikenal dengan teknologi USB, SATA, dan Wi-Fi menjadikan komunikasi serial sebagai teknik komunikasi yang mengalami perkembangan yang sangat cepat. 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. Dipandang cukup ekonomis hanya membutuhkan dua jalur komunikasi yaitu *transmit* (Tx) dan *receive* (Rx), dan jika menggunakan piranti kabel maka cukup membutuhkan minimal 3 kabel yaitu *transmit*, *receive*, dan *ground*. Prinsip komunikasi serial adalah pengiriman data secara serial dengan menggunakan karakter – karakter didalam ASCII. Karakter ini nantinya akan diubah menjadi sinyal digital oleh *hardware transmitter* (Tx), dan akan diterjemahkan lagi menjadi data karakter oleh *hardware receiver* (Rx). Komunikasi data serial mikrokontroler sangatlah sederhana, dikarenakan sudah memiliki instruksi – instruksi pemrograman yang standart.

Antarmuka kanal serial lebih kompleks atau sulit dibandingkan dengan antarmuka melalui kanal paralel, hal ini disebabkan karena:

1. Dari segi perangkat keras: adanya proses konversi data paralel menjadi serial atau sebaliknya menggunakan piranti tambahan yang disebut UART (*Universal Asynchronous Receiver/Transmitter*).
2. Dari segi perangkat lunak: lebih banyak register yang digunakan atau terlibat.

Namun di sisi lain antarmuka kanal serial menawarkan berapa kelebihan dibandingkan secara paralel, antara lain:

1. Kabel untuk komunikasi serial bisa lebih panjang dibandingkan dengan paralel: data - data dalam komunikasi serial dikirimkan untuk logika '1' sebagai tegangan -3 s/d -25 volt dan untuk logika '0' sebagai tegangan +3 s/d +25 volt, dengan demikian tegangan dalam komunikasi serial memiliki ayunan tegangan maksimum 50 volt, sedangkan pada komunikasi paralel hanya 5 volt. Hal ini menyebabkan gangguan pada kabel - kabel panjang lebih mudah diatasi dibandingkan pada paralel.
2. Jumlah kabel serial lebih sedikit: kita dapat menghubungkan dua perangkat komputer yang berjauhan dengan hanya 3 kabel untuk konfigurasi null modem, yaitu TXD (saluran kirim), RXD (saluran terima) dan *Ground*, bayangkan jika digunakan teknik paralel akan terdapat 20 – 25 kabel. Namun pada masing - masing komputer dengan komunikasi serial harus dibayar biaya antarmuka serial yang agak lebih mahal.
3. Banyaknya piranti saat ini (*palmtop, organizer, handphone*, dan lain - lain) menggunakan teknologi infra merah untuk komunikasi data, dalam hal ini pengiriman datanya dilakukan secara serial.
4. Untuk teknologi *embedded system*, banyak mikrokontroler yang dilengkapi dengan komunikasi serial (baik seri RISC maupun CISC) atau *Serial Communication Interface* (SCI): dengan adanya SCI yang terpadu pada 1C mikrokontroler akan mengurangi jumlah pin keluaran, sehingga hanya dibutuhkan 2 pin utama TxD dan RxD (di luar acuan *ground*).

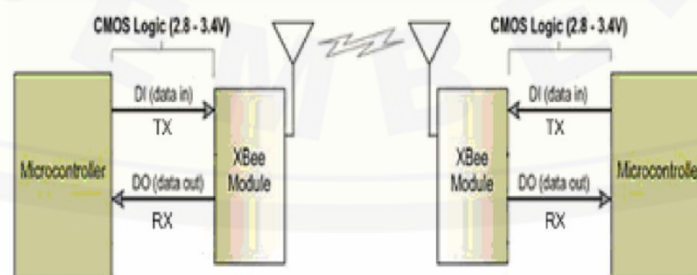
2.8 RF Module

Dalam perancangan *hardware* ini, komunikasi antara mikrokontroler dan komputer bersifat *wireless* sehingga diperlukan *RF module* yang berfungsi sebagai sistem transmisinya. Dalam perancangan komunikasi RF digunakan suatu modul *RF transceiver* YS1020. *RF module* ini telah diatur pada frekuensi 433.025 MHz dengan baudrate 19200 bps. Fungsi sistem RF diantaranya:

1. Penerimaan dan pemancaran HF (*High Frequency*).
2. Menyatukan data informasi dengan *signal* pembawa (Modulasi).
3. Memisahkan data informasi dengan *signal* pembawa (Demodulasi).
4. Pembangkit frekuensi untuk *signal* pemancaran dan penerimaan (*Frequency Synthesizer*).

Cara kerja sistem RF yaitu data informasi yang berasal dari sistem *Baseband* masih berbentuk getaran listrik yang berfrekuensi rendah, agar data informasi dapat dipancarkan dan diterima oleh *Base Station*, terlebih dahulu harus diproses oleh sistem RF agar menjadi getaran listrik yang berfrekuensi tinggi. Hal ini disebabkan frekuensi rendah tidak dapat membawa data informasi tersebut sampai jarak jauh. Hanya frekuensi tinggi (*High Frequency* = HF) atau Frekuensi Radio (*Radio Frequency* = RF) yang dapat membawanya ke seluruh penjuru dunia. Proses mengubah frekuensi rendah yang berupa data informasi menjadi frekuensi tinggi disebut bermodulasi. Getaran frekuensi tinggi yang dipancarkan *Base Station* dengan membawa sifat - sifat dan unsur - unsur frekuensi rendah dinamakan *Carrier Frequency* (frekuensi pembawa) atau frekuensi dukung.

Radio Frequency Transceiver atau pengirim dan penerima frekuensi radio ini berfungsi untuk komunikasi secara *full duplex*. Salah satu modul komunikasi *wireless* dengan frekuensi 2.4Ghz adalah XBee-PRO OEM ZigBee / IEEE 802.15.4.2.4 GHz. *Radio frequency transceiver* ini merupakan sebuah modul yang terdiri dari RF *receiver* dan RF *transmitter* dengan sistem *interface* serial UART *asynchronous*. (Rahman, 2014)



Gambar 2.10 Transceiver X-Bee Pro

(Sumber : XBee®/XBee-PRO® ZB RF Modules, Manual Book, hal: 27)

Gambar 2.11 Modul *wireless* X-Bee Pro(Sumber : Wikipidea, Modul *Wireless XBee Pro* dan cara kerja .com,2014)

2.8.1 Spesifikasi Modul *Wireless* X-Bee PRO

Tabel 2.1 Spesifikasi Modul X-Bee Pro

(Sumber : *Datasheet Xbee Pro XBP24BZ7WIT*)

<i>Platform</i>	XBee ZB	XBee- PRO ZB	Programmable XBee
<i>Performance</i>			
<i>RF Data Rate</i>	250 Kbps		
<i>Indoor /Urban Range</i>	133 ft (40m)	300 ft (90m)	
<i>Outdoor / RF Line-of-Sight Range</i>	400 ft (120m)	2 miles (3200m) / Intl 5000 ft (1500m)	
<i>Transmitt Power</i>	1.25 mW/2 mW	63 mW/Intl 10 Mw	
<i>Receiver Sensitivity</i>	-96 dBm in boost mode	-102 dBm	
<i>Features</i>			
<i>Adjustable Power</i>	Yes		
<i>I/O Interface</i>	3,3V CMOS UART,ADC,DIO	3,3V CMOS UART,SPI,PWM	
<i>Configuration Method</i>	API or AT command, local or over-the-air		
<i>Frequency Band</i>	2,4 GHz		
<i>Interference Immunity</i>	DSSS (Direct Sequence Spread Spectrum)		

<i>Serial Data Rate</i>	1200 bps-1 Mbps	
<i>ADC Input</i>	(4) 10-bit ADC inputs	
<i>Digital I/O</i>	10	
<i>Antenna Options</i>	Chip,Wire Whip, U.FL,RPSMA	PCB embedded Antenna,Wire Whip,U.FL
<i>Operating Temperature</i>	-40 C +85 C, 0-95%	

2.9 Packet Loss

Packet Loss, merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, dapat terjadi karena *collision* dan *congestion* pada jaringan dan hal ini berpengaruh pada semua aplikasi karena *retransmisi* akan mengurangi efisiensi jaringan secara keseluruhan meskipun jumlah *bandwidth* cukup tersedia untuk aplikasi-aplikasi tersebut. Umumnya perangkat jaringan memiliki *buffer* untuk menampung data yang diterima. Jika terjadi kongesti yang cukup lama, *buffer* akan penuh, dan data baru tidak akan diterima.

Beberapa penyebab terjadinya *packet loss* yaitu:

1. *Congestion*, disebabkan terjadinya antrian yang berlebihan dalam jaringan
2. *Node* yang bekerja melebihi kapasitas *buffer*
3. *Memory* yang terbatas pada *node*
4. *Policing* atau kontrol terhadap jaringan untuk memastikan bahwa jumlah trafik yang mengalir sesuai dengan besarnya *bandwidth*. Jika besarnya trafik yang mengalir didalam jaringan melebihi dari kapasitas *bandwidth* yang ada maka *policing control* akan membuang kelebihan trafik yang ada.
5. *Derau* atau yang biasa disebut *noise* adalah suatu sinyal gangguan yang bersifat akustik (suara), listrik, maupun elektronis yang hadir dalam suatu sistem (rangkaiian listrik/ elektronika) dalam bentuk gangguan yang bukan merupakan sinyal yang diinginkan.

Sumber *derau* dapat dikelompokkan dalam tiga kategori:

1. Sumber *derau intrinsic* yang muncul dari fluktuasi acak di dalam suatu sistem fisik seperti *thermal* dan *shot noise*.
2. Sumber *derau* buatan manusia seperti motor, *switch*, elektronika digital.
3. *Derau* karena gangguan alamiah seperti petir dan bintik matahari.

Perhitungan *packet loss* dilakukan dengan cara membandingkan data yang dikirim oleh komputer di kurangi data yang diterima oleh komputer, kemudian dihitung persentase data yang hilang, dapat dihitung dengan rumus : (sumber : Bayu, 2014)

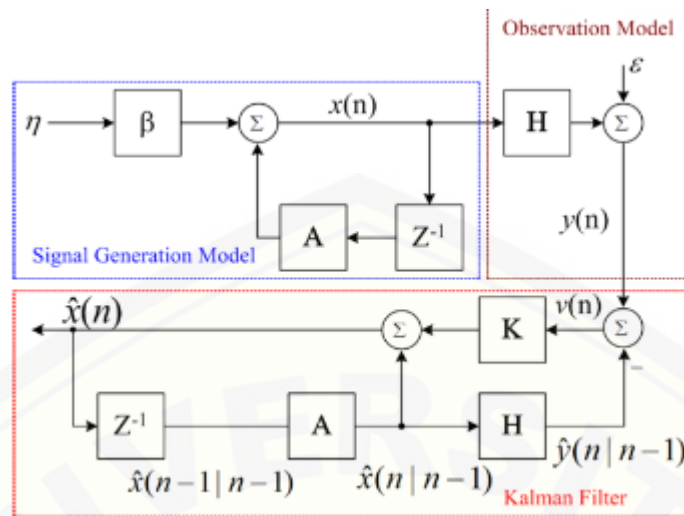
$$\text{Packet loss \%} = \frac{|DT-DD|}{DT} \times 100\% \dots\dots\dots(1)$$

Dimana : DT = *Packet Data* Dikirim
 DD = *Packet Data* Diterima

2.10 Kalman Filter

Salah satu dari beberapa masalah dasar dalam teori komunikasi adalah proses *filtering* terhadap sinyal *noise* untuk menghasilkan sinyal asli yang lebih bersih. Pada umumnya sinyal asli dapat berupa tegangan, arus, posisi, kecepatan, percepatan, dan kombinasi lainnya.

Kalman filter merupakan salah satu metode untuk mengestimasi masalah yang kompleks dari proses yang tidak stasioner berdasarkan pada pendekatan ruang keadaannya (*statespace*). *Kalman filter* dapat mengestimasi sinyal yang menyimpang berdasarkan metode *least square error rekursif* dan biasanya disebut estimator yang rekursif yang berarti hasil pengukuran pada keadaan sekarang dan sebelumnya dibutuhkan untuk menghitung estimasi pada keadaan sekarang. Metode ini diperkenalkan pertama kali oleh Rudolph E. Kalman (1960).



Gambar 2.12 Struktur Kalman Filter

(Sumber : Supeno, 2012)

Persamaan *state space* menggambarkan proses dinamik dari sebuah sinyal. Dari struktur *Kalman filter* pada gambar 2.11 dapat diturunkan model persamaan observasi serta *noise* dari sinyal observasi ditunjukkan pada persamaan :

$$x(n) = A x(n-1) + \beta \eta \dots\dots\dots(2)$$

$$y(n) = Hx(n) + \epsilon \dots\dots\dots(3)$$

Dimana x adalah sinyal generasi, y adalah sinyal observasi yang mengandung *noise*, A , H , η dan ϵ adalah matrik dari model sinyal, matrik dari model observasi, *noise* eksitasi dan *noise* pengukuran. Sedangkan untuk penurunan model persamaan *Kalman filter* untuk mengestimasi nilai yang benar dari sinyal dengan persamaan :

$$x(n) = x(n | n-1) + K(y(n) - Hx(n | n-1)) \dots\dots\dots(4)$$

2.10.1 Algoritma Kalman Filter

Algoritma *Kalman filter* yang digunakan untuk mengestimasi sinyal yang menyimpang sehingga didapatkan hasil pengukuran yang benar adalah sebagai berikut : (sumber: <https://www.youtube.com/watch?v=biY7F-tLwE8>)

1. *Predict Next State*

$$x(t) = F * x(t-1) + B * U$$

2. *Predict Next Covariance*

$$P(t) = F * P(t-1) * F^T + Q$$

3. *Compute The Kalman Gain*

$$K = \frac{P(t) * H^T}{(H * P(t) * H^T + R)}$$

4. *Update The State Estimate*

$$x(t) = x(t) + K * (\text{measurement}(t) - H * x(t))$$

5. *Update Covariance Estimate*

$$P(t) = (I - K * H) * P(t)$$

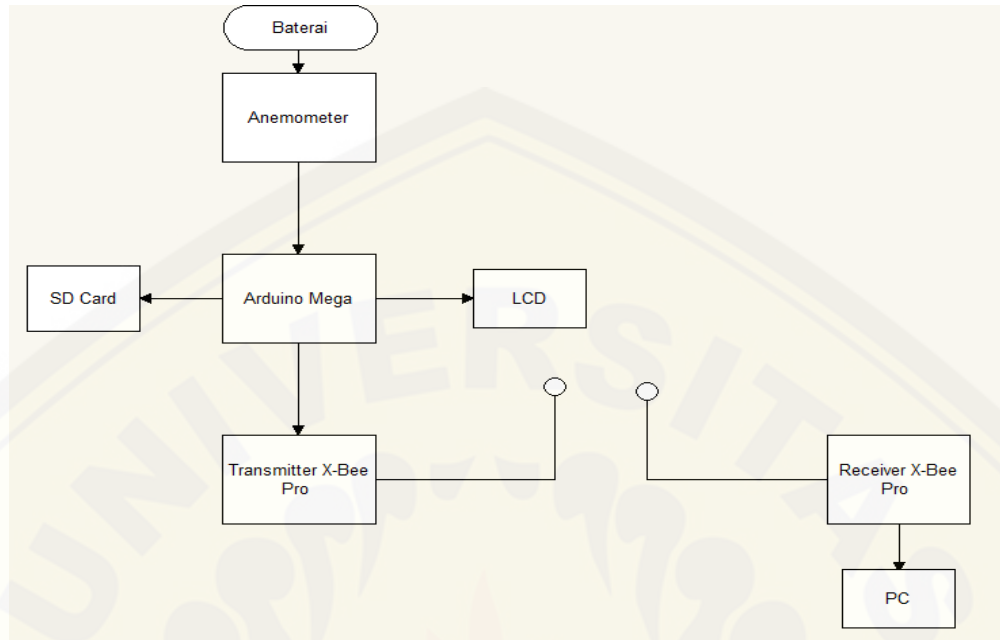
BAB 3. METODOLOGI PENELITIAN

3.1 Prosedur Penelitian

Dalam pembuatan skripsi dan penelitian ini, dibuat langkah-langkah atau prosedur penelitian sebagai berikut :

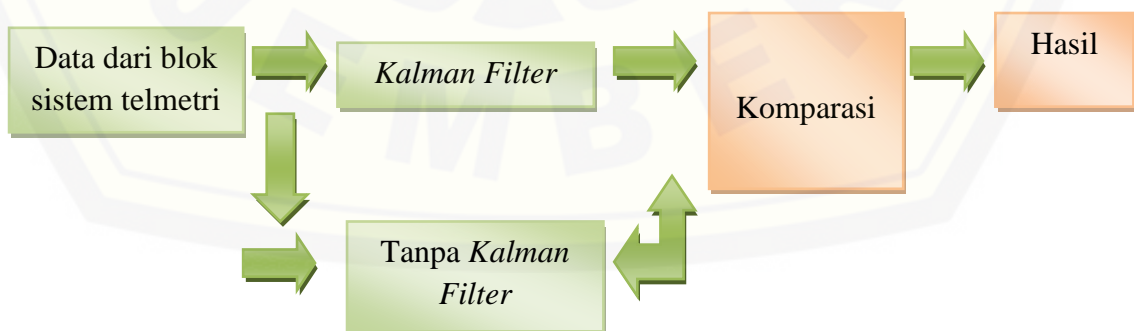
- a. Tahap Persiapan
Persiapan dengan mengurus segala keperluan.
- b. Studi literatur terhadap obyek yang akan dikerjakan.
Studi literatur ini bertujuan untuk menambah sumber dan metode yang akan digunakan.
- c. Perancangan mekanik dan *hardware*.
Perancangan mekanik dan *hardware* terdiri dari disain mekanik sistem sensor.
- d. Pembuatan dan pengujian *hardware*.
Disain mekanik dan *hardware* yang telah dibuat kemudian dilakukan pengujian dengan cara menguji masing-masing rangkaian sebelum direalisasikan menjadi sebuah sistem sensor.
- e. Pembuatan *software*.
Pada tahap ini adalah pembuatan *software* pengolah sinyal untuk mendapatkan data hasil pengukuran kecepatan angin yang benar dengan menggunakan *Kalman Filter*.
- f. Pengolahan data
Setelah mengambil data maka data tersebut di analisis bagaimana kualitas data kecepatan angin yang sampai pada PC.

3.2 Blok Sistem



Gambar 3.1 Blok Sistem

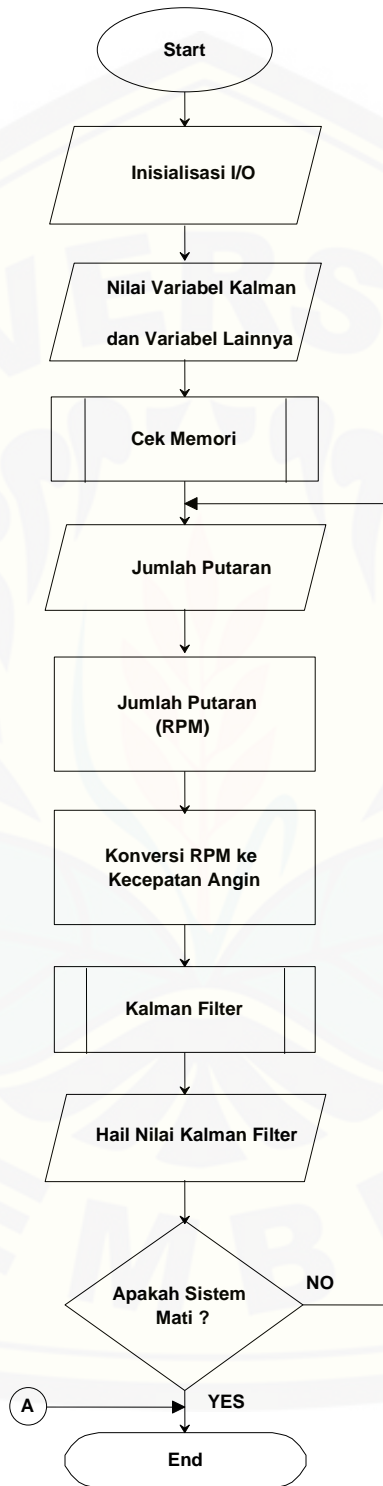
Pada blok sistem diatas menjelaskan tentang perancangan sistem secara keseluruhan. Dimana anemometer dan X-Bee Pro dipihak *Transmitter* (Tx) di *supply* menggunakan baterai. Pada pihak *transmitter* (Tx) data tersebut secara otomatis akan tersimpan didalam SD Card karena menggunakan sistem *data logger* dan juga ditampilkan menggunakan LCD. Setelah didapat nilai kecepatan angin maka data dikirimkan secara *wireless*. Data tersebut diterima menggunakan X-Bee Pro *Receiver* dan ditampilkan ke PC.



Gambar 3.2 Blok Sistem Kalman Filter

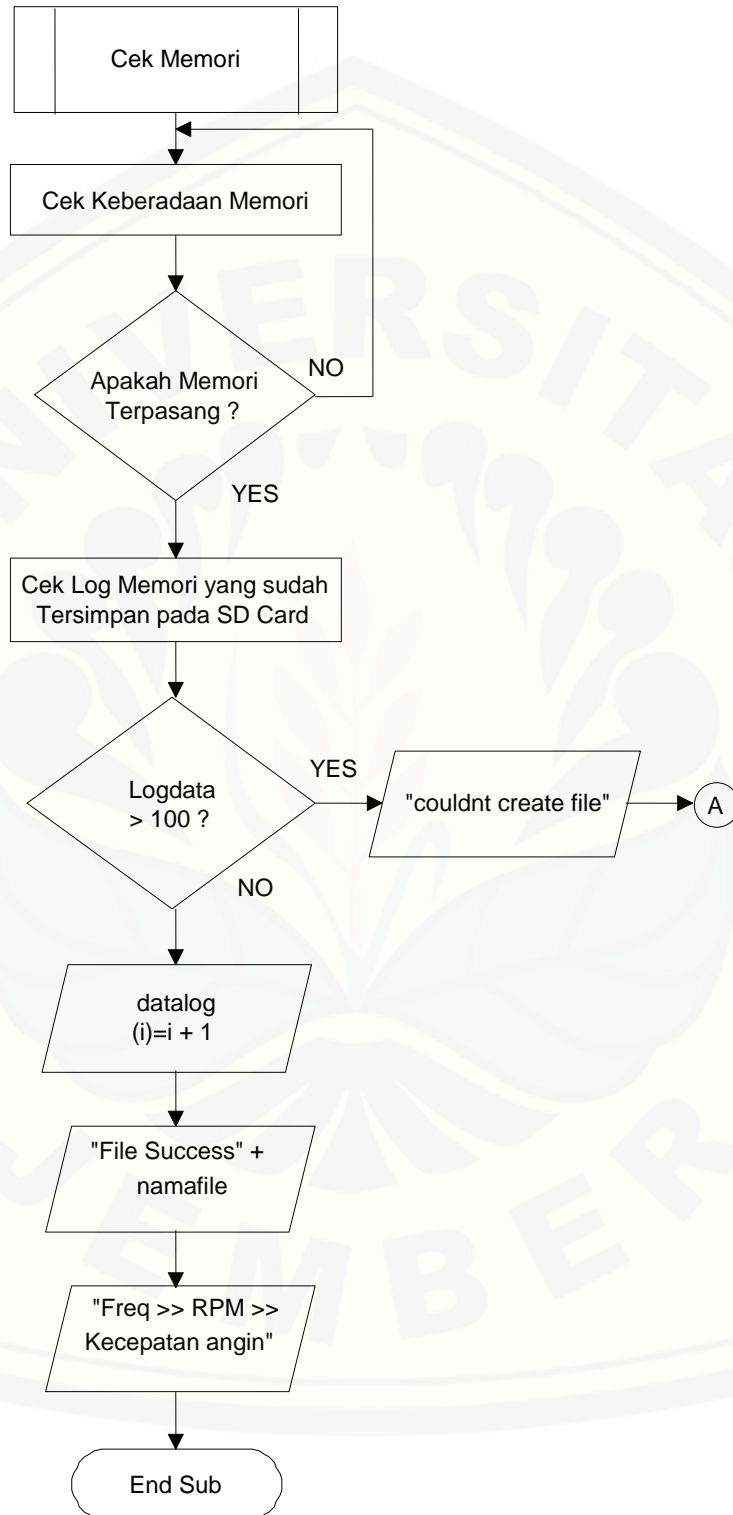
Pada blok sistem ini menjelaskan bahwa alat pengukuran kecepatan angin menggunakan sistem telemetri. Perancangan ini juga dilengkapi dengan modul RF sebagai media untuk *transfer* data juga dilengkapi perekam data (*data logger*). *Data logger* menyimpan data teknis dan sensor. Setelah alat pengukur kecepatan angin mulai mengukur maka data secara otomatis akan dikirimkan dari sisi *transmitter* (Tx) ke sisi *receiver* (Rx). Data yang diperoleh bukan hanya kecepatan angin saja, namun kecepatan angin tersebut dilakukan proses *filter* menggunakan *Kalman Filter*. Sehingga yang akan tampil di LCD merupakan data kecepatan angin dan kecepatan yang telah dilakukan *filter* menggunakan *Kalman Filter*. *Monitoring* kecepatan angin pada pihak *receiver* (Rx) atau PC menggunakan *software* Visual Basic. Dari situ dibandingkan hasil dari data yang dilakukan *filter* menggunakan *Kalman Filter* dengan data kecepatan angin yang asli.

3.3 Flowchart Utama



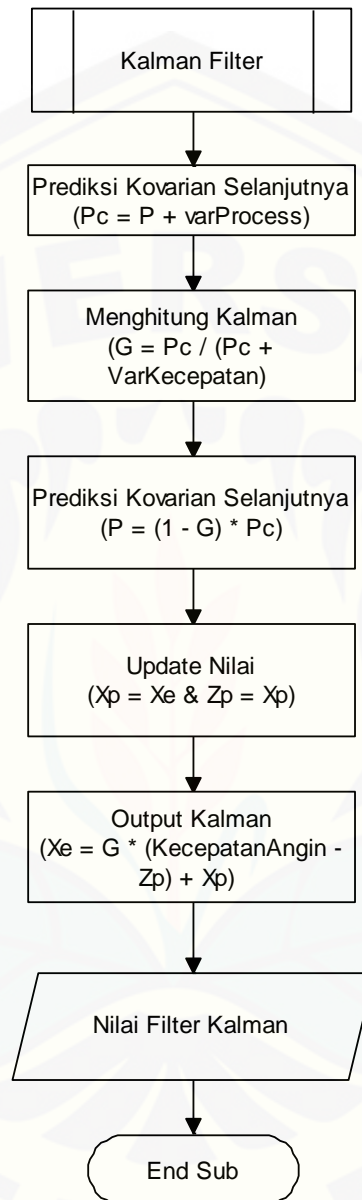
Gambar 3.3 Flowchart Utama

3.4 Flowchart Subrutin Cek Memori



Gambar 3.4 Flowchart Subrutin Cek Memori

3.5 Flowchart Subrutin Kalman Filter



Gambar 3.3 Flowchart Subrutin Kalman Filter

BAB 4. HASIL DAN PEMBAHASAN

4.1 Pengujian Program *Kalman Filter*

Pada penelitian ini perancangan program *Kalman Filter* menggunakan *software* Arduino 1.6.5.



```

anemometer3 | Arduino 1.6.5

anemometer3

//float varKecepatan = 0.024537879;
//float varKecepatan = 0.058788479;
float varKecepatan = 1.136115;
float varProcess = 1e-8;
float Pc = 0.0;
float G = 0.0;
float P = 1.0;
float Xp = 0.0;
float Zp = 0.0;
float Xe = 0.0;

int Phi = 3.1428571428571428571428571428571;
int JariJari = 1.8;
float frekuensi;
// for the data logging shield, we use digital pin 10 for the SD cs line
const int chipSelect = 53;
// the logging file
File logfile;
void error(char *str)
{
  lcd.setCursor(0,0);
  lcd.print("PASANG MEMORI...!");
  lcd.setCursor(1,1);
  lcd.print("LOG DATA GAGAL");
  Serial.println(str);
  // red LED indicates error
  digitalWrite(redLEDPin, HIGH);
}

while(1);

```

Gambar 4.1 Program *Kalman Filter* pada *software* Arduino 1.6.5

float varKecepatan = 1.136115;

Untuk mendapatkan nilai diatas maka kita perlu mengambil nilai variasi terlebih dahulu pada sensor untuk dapat ditentukan variasi variabel kecepatannya. Ini dapat kita lakukan dengan cara memutar alat beberapa saat dengan kecepatan putar yang konstan untuk mendapatkan nilai variasinya. Seperti yang ditunjukkan pada *Microsoft Excel* kalibrasi, dimana dengan kecepatan putar yang konstan

ternyata nilai yang dihasilkan sensor tidak konstan, sehingga dengan mengacu pada nilai variasi tersebut kita dapat menggunakan rumus $G5 = \text{VAR}(C4:C114)$ untuk mendapatkan nilai variasinya, sehingga didapatkan nilai 1.136115 sebagai nilai variasinya.

Dibawah ini adalah variabel variasi yang diasumsikan dimana semakin kecil nilainya maka semakin *steady* hasil *filter* atau semakin halus tidak mengikuti nilai masukan dari sensor.

float varProcess = 1e-8;

Contoh jika 1e-8 diganti dengan 1e-9 maka yang terjadi adalah grafik sinyal keluaran akan semakin tidak sesuai dengan nilai masukan sensor, namun jika semakin besar nilainya yaitu 1e-7 maka sinyal keluaran akan semakin menyesuaikan dengan nilai masukan sensor yang nilainya tidak *steady*, alhasil pada grafik sinyal keluaran akan naik turun atau tidak *steady* (halus).

Variabel untuk kovarian selanjutnya

float Pc = 0.0;

Mewakili lambang K fungsi variabel di sini adalah variabel tumpang *kalman gain*, sehingga disingkat G.

float G = 0.0;

Variabel tumpang kovarian estimasi

float P = 1.0;

Variabel tumpang biasa, penamaan bebas. Fungsi dari variabel ini adalah berfungsi memindahkan nilai agar nilai di dalam variabel tidak berubah saat proses *Kalman Filter* terjadi

float Xp = 0.0;

float Zp = 0.0;

float Xe = 0.0;



```

anemomoter3 | Arduino 1.6.5

anemomoter3

Pc = P + varProcess;
G = Pc / (Pc + varKecepatan);
P = (1 - G) * Pc;
Xp = Xe;
Zp = Xp;
Xe = G * (KecepatanAngin - Zp) + Xp;

Serial.print(KecepatanAngin);
Serial.print("#");
Serial.println(Xe);

logfile.print(" ");
logfile.print(count);
logfile.print(" ");
logfile.print(RPM);
logfile.print(" ");
logfile.print(KecepatanAngin);
logfile.print(" ");
logfile.print(Xe);

#if ECHO_TO_SERIAL
Serial.print(" ");
Serial.print(count);
#endif // ECHO_TO_SERIAL

```

Gambar 4.2 Rumus *Kalman Filter* pada *software* Arduino 1.6.5

Memprediksi kovarian selanjutnya (*next covariance*)

$$\mathbf{Pc} = \mathbf{P} + \mathbf{varProcess};$$

Menghitung *Kalman Filter*

$$\mathbf{G} = \mathbf{Pc} / (\mathbf{Pc} + \mathbf{varKecepatan});$$

Memperbarui kovarian estimasi atau memprediksikan kovarian selanjutnya (*update the covariance estimate*)

$$\mathbf{P} = (\mathbf{1} - \mathbf{G}) * \mathbf{Pc};$$

Update nilai sebelumnya

$$\mathbf{Xp} = \mathbf{Xe};$$

Update nilai sebelumnya

$$\mathbf{Zp} = \mathbf{Xp};$$

Kalman Filter memperbarui kondisi estimasi nilai yang keluar dari sensor (nilai *output Kalman Filter*)

$$\mathbf{X}_e = \mathbf{G} * (\text{KecepatanAngin} - \mathbf{Z}_p) + \mathbf{X}_p;$$

4.2 Pengujian Program

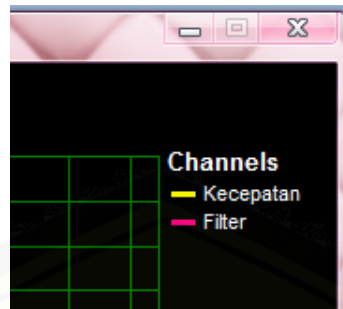
4.2.1 Pengujian Tampilan Pada *Software*

Pada penelitian ini menggunakan *software* Visual Basic di sisi penerima (Rx) untuk *monitoring* kecepatan angin serta kecepatan yang telah dilakukan proses *Kalman Filter*. Jika ingin memulai pengukuran pastikan bahwa modul X-Bee Pro pada sisi penerima (Rx) yang terhubung dengan PC sudah terpasang. Tampilan awal ketika *software user interface* yang digunakan untuk *monitoring* kecepatan angin dapat dilihat pada gambar 4.3 berikut ini :



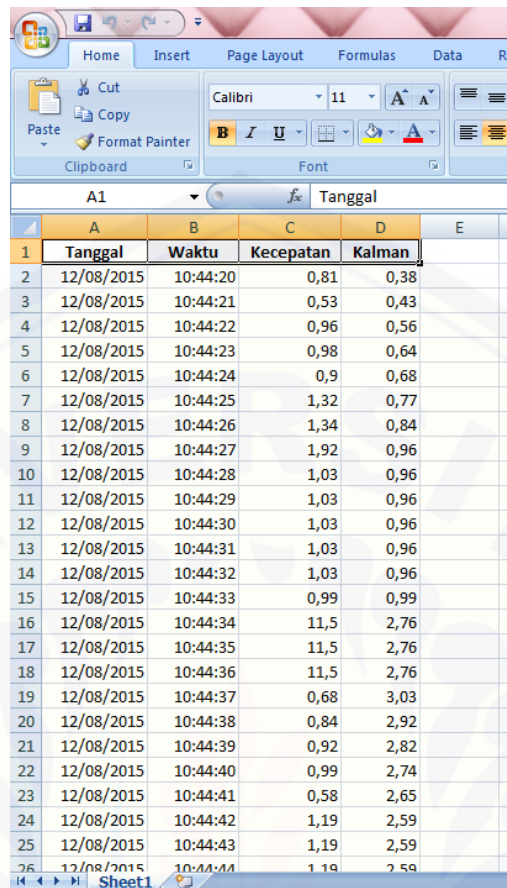
Gambar 4.3 Tampilan *user interface software* Visual Basic

Software ini menampilkan informasi seperti jam, tanggal, nilai kecepatan angin, nilai kecepatan angin yang telah dilakukan proses *filter* menggunakan *Kalman Filter*, dan grafik antara kecepatan angin dengan kecepatan angin yang telah dilakukan proses *filter* menggunakan *Kalman Filter*. Dipojok kanan terdapat keterangan untuk *channel* yang berwarna kuning merupakan kecepatan dan yang berwarna merah muda adalah hasil *Kalman Filter*.



Gambar 4.4 Keterangan *channel* pada grafik

Grafik pada *software* tersebut dilakukan secara terus menerus selama *wireless* penerima dalam kondisi *connect*. Menu yang disediakan pada *software* ini antara lain *com port* sebagai pengaturan pemilihan *port serial*, *baud rate* sebagai menu pengaturan *baud rate* yang digunakan, *time sampling* (mS) sebagai menu pengaturan pengambilan data pada grafik berdasarkan waktu yang dapat ditentukan oleh *user* dengan satuan *milisecond* (mS) dan *save to excel file* sebagai menu pengaturan pengambilan data yang disimpan pada *Microsoft excel* berdasarkan waktu yang ditentukan oleh *user* dengan satuan *millisecond* (ms). Menu *connect* digunakan untuk pengambilan data kecepatan angin yang dikirimkan oleh sisi pengirim (Tx), sedangkan menu *disconnect* digunakan untuk memutuskan pengiriman data kecepatan angin. Kemudian mengatur *com port* yang sesuai. Selanjutnya *baud rate* yang digunakan yaitu 9600 karena pada nilai tersebut data dapat dikirim dan diterima dengan baik. Kemudian mengatur waktu pengambilan data pada menu *time sampling* dan *save to excel*. Pada pengujian yang telah dilakukan menggunakan pengambilan data tiap 1000 mS atau 1 detik sekali. Kemudian pilih *connect* untuk menghubungkan program aplikasi tersebut dengan *wireless* yang digunakan. Untuk mengetahui hasil penyimpanan data pada *Microsoft excel* dapat dilakukan dengan membuka *file excel* yang telah dipilih pada langkah awal ketika pertama kali membuka program aplikasi tersebut.

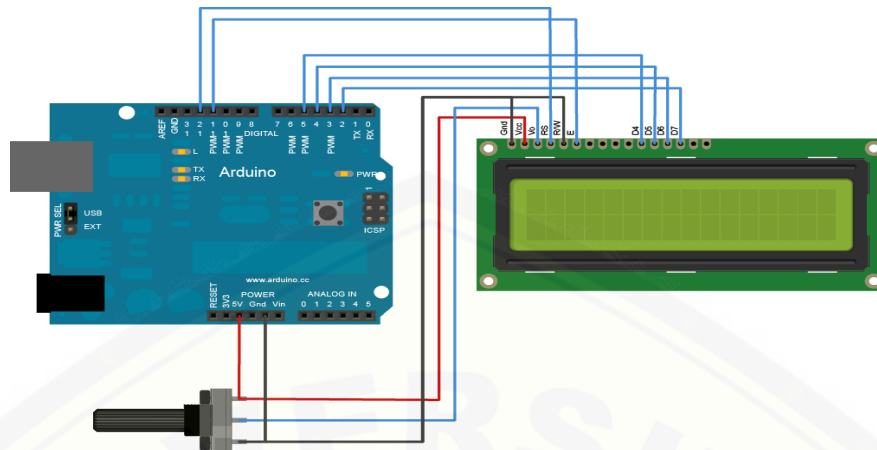


	A	B	C	D	E
1	Tanggal	Waktu	Kecepatan	Kalman	
2	12/08/2015	10:44:20	0,81	0,38	
3	12/08/2015	10:44:21	0,53	0,43	
4	12/08/2015	10:44:22	0,96	0,56	
5	12/08/2015	10:44:23	0,98	0,64	
6	12/08/2015	10:44:24	0,9	0,68	
7	12/08/2015	10:44:25	1,32	0,77	
8	12/08/2015	10:44:26	1,34	0,84	
9	12/08/2015	10:44:27	1,92	0,96	
10	12/08/2015	10:44:28	1,03	0,96	
11	12/08/2015	10:44:29	1,03	0,96	
12	12/08/2015	10:44:30	1,03	0,96	
13	12/08/2015	10:44:31	1,03	0,96	
14	12/08/2015	10:44:32	1,03	0,96	
15	12/08/2015	10:44:33	0,99	0,99	
16	12/08/2015	10:44:34	11,5	2,76	
17	12/08/2015	10:44:35	11,5	2,76	
18	12/08/2015	10:44:36	11,5	2,76	
19	12/08/2015	10:44:37	0,68	3,03	
20	12/08/2015	10:44:38	0,84	2,92	
21	12/08/2015	10:44:39	0,92	2,82	
22	12/08/2015	10:44:40	0,99	2,74	
23	12/08/2015	10:44:41	0,58	2,65	
24	12/08/2015	10:44:42	1,19	2,59	
25	12/08/2015	10:44:43	1,19	2,59	
26	12/08/2015	10:44:44	1,19	2,59	

Gambar 4.5 Data yang disimpan dalam *Microsoft excel*

4.2.2 Pengujian LCD

Pada penelitian ini dilakukan pengujian pada LCD yang berfungsi untuk melihat proses dalam pengujian sensor *optocoupler* untuk menampilkan data kecepatan angin. Untuk mengetahui kinerja dari LCD tersebut maka dilakukan pengujian dengan menghubungkan pin – pin digital modul arduino yaitu pin12, pin 11, pin 5, pin 4, pin 3 dan pin 2 dan Vcc dan Gnd, untuk menampilkan data yang berupa kecepatan angin menggunakan LCD *display* 16x2 pada alat ini. Gambar di bawah merupakan gambar rangkaian antara modul arduino yang dikoneksikan dengan LCD.



Gambar 4.6 Wiring Diagram LCD

(Sumber : Bhakti,2014)

Untuk menguji digunakan LCD *library* yang terdapat pada *software* Arduino 1.6.5 program tersebut lalu di *load* ke modul yang sudah terkoneksi dengan LCD, berikut ini merupakan tampilan dari *library* LCD yang terdapat pada *software* Arduino 1.6.5

```

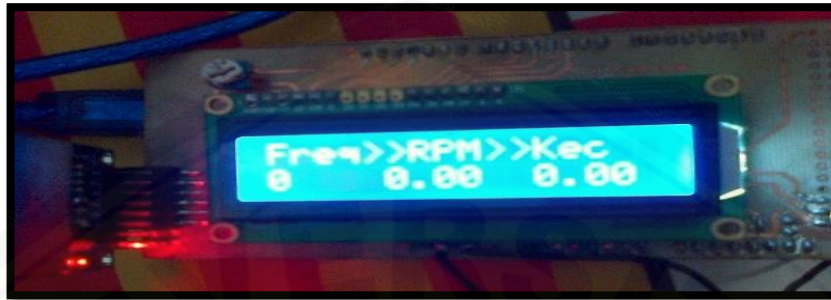
anemometer3 | Arduino 1.6.5
anemometer3
** MISO - pin 12 on Arduino Uno/Duemilanove/Diecimila
** CLK - pin 13 on Arduino Uno/Duemilanove/Diecimila
** CS - depends on your SD card shield
or module.
*/
#include <FreqCount.h>
#include <SPI.h>
// include the SD library:
#include "LiquidCrystal.h";
LiquidCrystal lcd(12, 11, 5, 4, 3, 2 );
#include <SD.h>
// how many milliseconds between grabbing data and logging it. 1000 ms is once a second
#define LOG_INTERVAL 1000 // mills between entries (reduce to take more/faster data)
// how many milliseconds before writing the logged data permanently to disk
// set it to the LOG_INTERVAL to write each time (safest)
// set it to 10*LOG_INTERVAL to write all data every 10 datareads, you could lose up to
// the last 10 reads if power is lost but it uses less power and is much faster!
#define SYNC_INTERVAL 1000 // mills between calls to flush() - to write data to the card
uint32_t syncTime = 0; // time of last sync()
// the digital pins that connect to the LEDs
#define redLEDPin 1
#define buzzer 9

//float varKecepatan = 0.024537879;
//float varKecepatan = 0.058788479;
float varKecepatan = 1.136115;
float varProcess = 1e-8;
float Pc = 0.0;
float G = 0.0;
float P = 1.0;

```

Gambar 4.7 Library LCD pada Arduino 1.6.5

Lalu program tersebut di *load* ke modul dan akan menghasilkan karakter kata dalam LCD yang diinginkan.



Gambar 4.8 Tampilan LCD

4.3 Pengujian *Wireless*

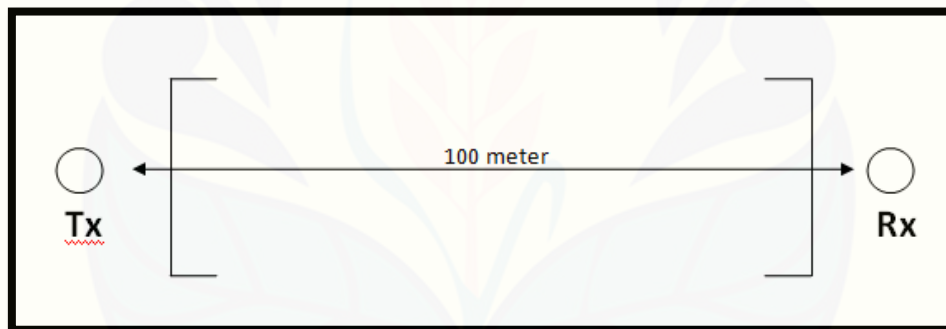
Pengujian *wireless* ini dilakukan untuk mengetahui bentuk pengiriman data yang ideal untuk mengantisipasi terjadinya kesalahan pada saat pengiriman. Sehingga diperoleh berapa jumlah pengiriman yang sesuai agar perintah dapat terkirim dengan baik. Selain itu pengujian kepekaan sensor *Optocoupler*. Pengujian jarak juga diperlukan untuk mengetahui jarak maksimum yang dapat dijangkau alat pengukur kecepatan angin ini untuk mengirimkan data. Pengujian *wireless* disini akan dilakukan dalam dua kondisi yaitu kondisi tanpa halangan (*loss space*) yang pengujiannya dilakukan di lapangan sepak bola dan kondisi ada halangan (*obstacle*) yang pengujiannya dilakukan di antara rumah dan halaman. Data tersebut akan ditampilkan menggunakan *software* Visual Basic secara *real time*.

4.3.1 Pengujian *Wireless* Kondisi *Loss Space*

Pada penelitian ini akan dilakukan pengambilan data dalam dua kondisi, yaitu pada kondisi *loss space* dan *obstacle*. Tujuannya agar dapat mengetahui pengaruh dua kondisi tersebut pada pengiriman data dari alat pengukur kecepatan angin. Pengambilan data dalam kondisi *loss space* ini dilakukan di lapangan terbuka. Dalam pengujian kondisi *loss space* diambil 5 *sample* jarak mulai dari jarak maksimal yaitu 100 meter, 80 meter, 60 meter, 40 meter, dan 20 meter. Data yang

didapat nanti adalah data kecepatan angin dan data kecepatan angin setelah di *filter* menggunakan *Kalman Filter*. Alat pengukur kecepatan angin ini dilengkapi dengan *data logger* yang nanti otomatis data akan disimpan didalam SD card. Tujuannya nilai pada *data logger* ini sebagai nilai patokan kecepatan yg diterima oleh sensor secara langsung dan dibandingkan dengan data yang dikirim melalui *wireless*. Setelah data diambil maka akan dihitung nilai *packet loss* masing-masing pengukuran dan nilai rata-rata *packet loss* keseluruhan menggunakan rumus matematis *packet loss*.

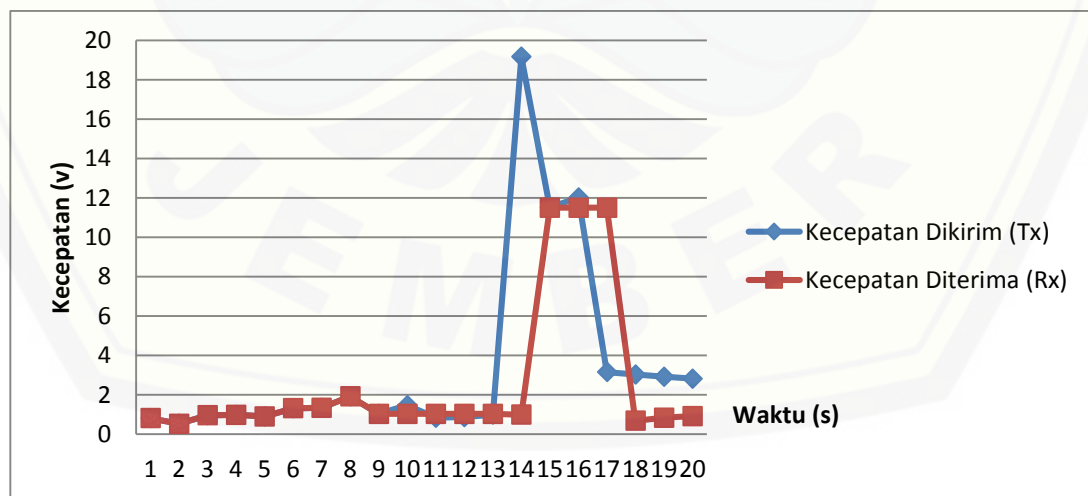
Pada pengambilan data pertama dalam kondisi *loss space* dilakukan pada jarak maksimal yaitu 100 meter dimana setiap pengukuran akan diambil sebanyak 20 data secara *realtime*. Pengambilan data tersebut setiap detik, sehingga jika 20 data maka membutuhkan waktu hanya 20 detik setiap pengukuran. Berikut merupakan kondisi pengujian *loss space* pada jarak maksimal yaitu 100 meter.



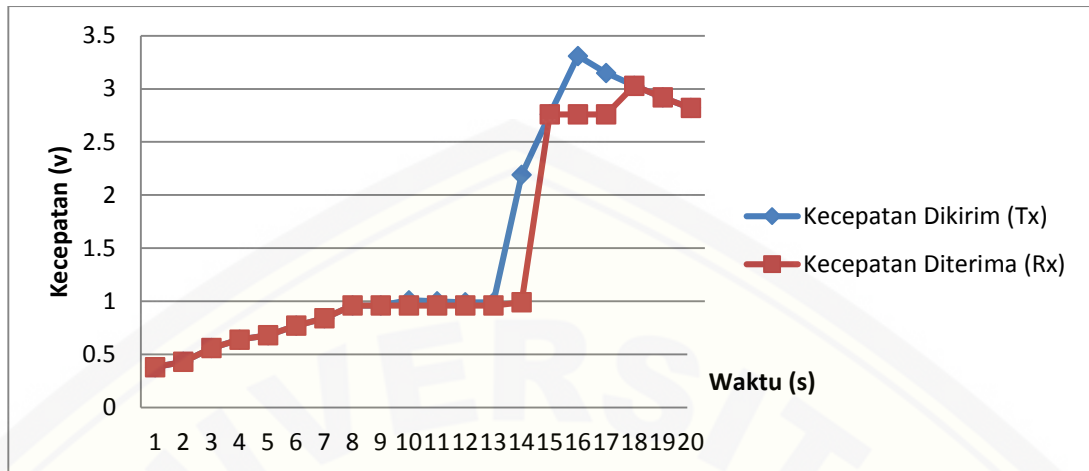
Gambar 4.9 Pengambilan data kondisi *loss space* pada jarak 100 meter



Pada tabel 4.1 merupakan tabel hasil pengujian *wireless* pada kondisi *loss space* pada jarak maksimal yaitu 100 meter. Dimana pengirim (Tx) berada dibelakang gawang 1 dan penerima (Rx) berada dibelakang gawang 2 seperti terlihat pada gambar 4.9 diatas. Pada tabel 4.1 terlihat bahwa terdapat data kecepatan angin yang asli atau yang berasal dari *data logger* dan data kecepatan angin yang diterima. Serta terdapat juga nilai kecepatan angin yang telah dilakukan *filter* menggunakan *Kalman Filter* pada sisi pengirim (Tx) dan sisi penerima (Rx). Dari 20 data yang didapat terlihat hampir seluruh data yang dikirim sebagian besar sama dengan data yang diterima. Dapat dilihat grafiknya pada gambar 4.10 dan gambar 4.11 merupakan grafik perbandingan antara kecepatan yang dikirim dan diterima tanpa *Kalman Filter* dan grafik antara kecepatan dikirim dan yang diterima yang menggunakan *Kalman Filter*. Dengan jumlah nilai rata – rata *packet loss* keseluruhan untuk kecepatan tanpa *Kalman Filter* sebesar 0,32 % dan untuk rata – rata *packet loss* keseluruhan kecepatan yang menggunakan *Kalman Filter* sebesar 0,048 %. Hal ini membuktikan bahwa modul *wireless* X-Bee Pro yang digunakan dapat mengirimkan data dengan baik pada saat *wireless* masih dalam rentang jarak jangkauan maksimalnya. Sedangkan ada beberapa data yang tidak sesuai antara data yang dikirim dan data diterima antara lain pada data nomor 10 , 11, 12, 13, 14, 16, 17, 18, 19, 20 (pada tabel 4.1).



Gambar 4.10 Grafik kecepatan angin dikirim (Tx) dan diterima (Rx)



Gambar 4.11 Hasil *Kalman Filter* dikirim (Tx) dan diterima (Rx)

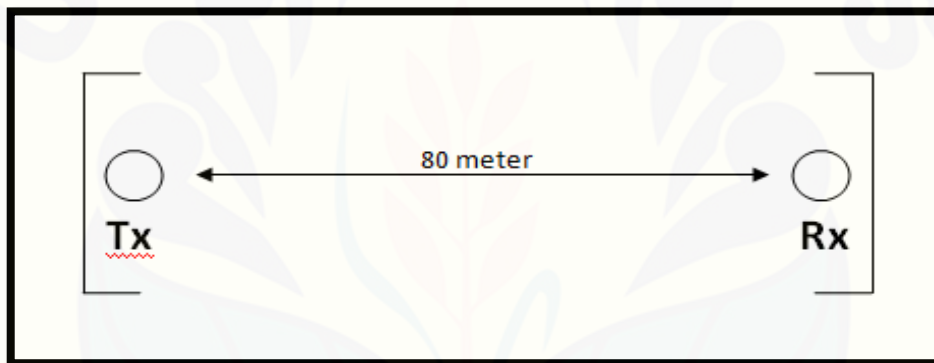


Gambar 4.12 Grafik kecepatan dan *Kalman Filter* pada jarak 100 meter

Disini *Kalman Filter* berfungsi untuk mengurangi *noise* agar tidak mengganggu pengiriman data. Ketika kecepatan angin dalam keadaan stabil contohnya pada data nomor 1 sampai 5 yaitu kecepatan angin stabil di nilai 0, maka nilai dari *Kalman Filter* juga akan stabil di nilai 0, ketika kecepatan angin naik menjadi 1 maka nilai dari *Kalman Filter* nilainya akan naik perlahan begitu selanjutnya jika kecepatan terus naik sampai nilainya 19. Begitu juga jika nilai kecepatan angin ternyata turun contohnya pada data ke 16 dan 17, maka nilai

Kalman Filter juga akan mengikuti turun. Apabila terjadi kenaikan atau penurunan nilai kecepatan angin yang drastis nilai *Kalman Filter* akan mengikuti kenaikan atau penurunannya secara perlahan. Dapat dilihat pada grafik pada gambar 4.12. Dengan *Kalman Filter* ini data yang naik turun tadi dibuat stabil. Dimana garis yang berwarna kuning merupakan kecepatan, dan garis yang berwarna merah muda merupakan *Kalman Filter*.

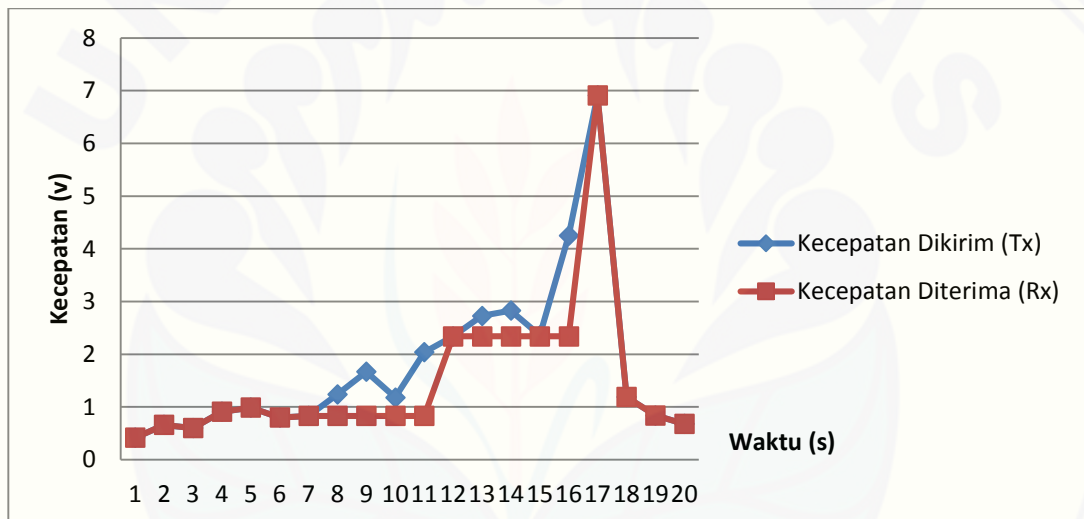
Pengambilan data kedua dengan jarak 80 meter dalam kondisi *loss space* dimana setiap pengukuran akan diambil sebanyak 20 data secara *realtime*. Pengambilan data tersebut setiap detik, sehingga jika 20 data maka membutuhkan waktu hanya 20 detik setiap pengukuran. Berikut merupakan kondisi pengujian *loss space* pada jarak 80 meter.



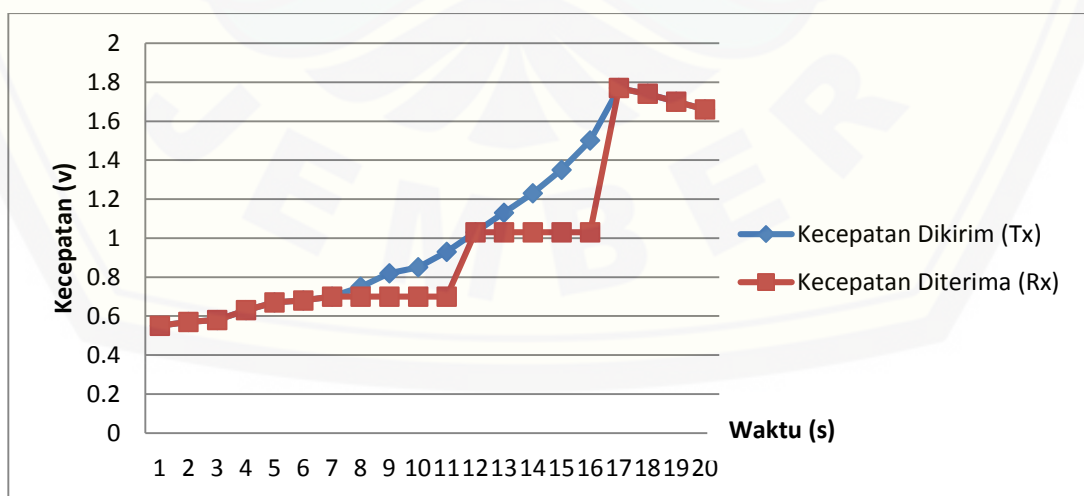
Gambar 4.13 Pengambilan data kondisi *loss space* pada 80 meter



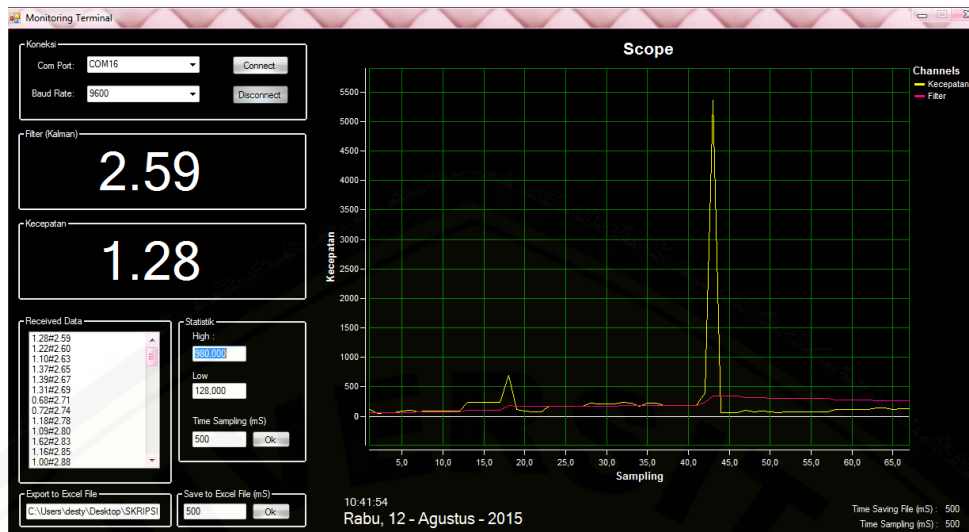
Tabel 4.2 merupakan data hasil pengujian *wireless* dalam kondisi *loss space* pada jarak 80 meter. Sama halnya dengan pengukuran pada jarak 100 meter, masih terdapat selisih antara data yang dikirim dan diterima. Dapat dilihat grafiknya pada gambar 4.14 dan gambar 4.15 merupakan grafik perbandingan antara kecepatan yang dikirim dan diterima tanpa *Kalman Filter* dan grafik antara kecepatan dikirim dan yang diterima yang menggunakan *Kalman Filter*. Setelah diperoleh nilai *packet loss* masing- masing kemudian dihitung nilai rata – rata *packet loss* keseluruhan dan diketahui nilai rata – rata *packet loss* keseluruhan untuk kecepatan sebesar 0,12 % dan untuk rata – rata *packet loss* keseluruhan kecepatan yang menggunakan *Kalman Filter* sebesar 0,06 %.



Gambar 4.14 Grafik kecepatan angin dikirim (Tx) dan diterima (Rx)

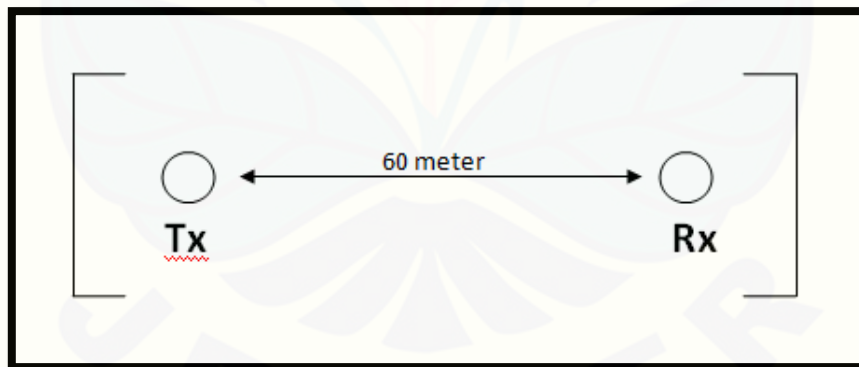


Gambar 4.15 Hasil *Kalman Filter* dikirim (Tx) dan diterima (Rx)



Gambar 4.16 Grafik kecepatan dan *Kalman Filter* pada jarak 80 meter

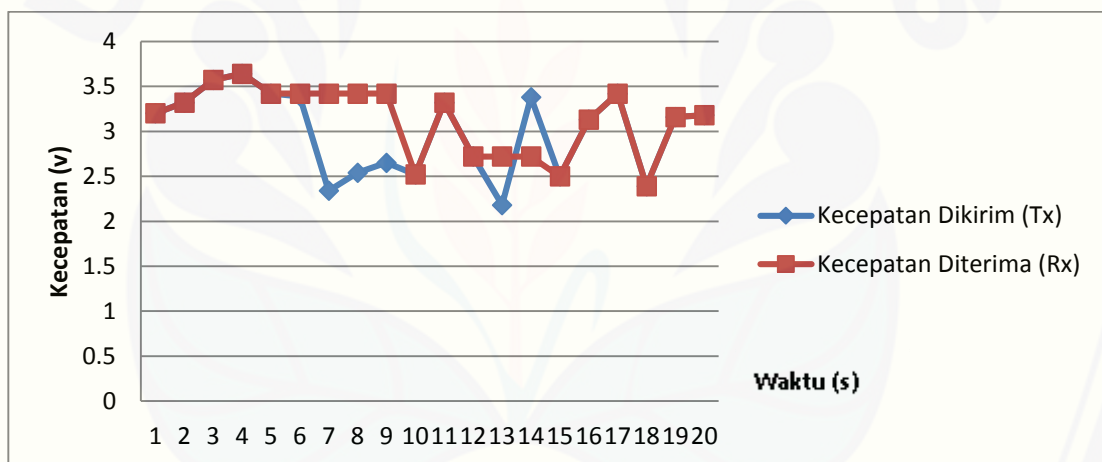
Pengambilan data ketiga dengan jarak 60 meter dalam kondisi *loss space* dimana setiap pengukuran akan diambil sebanyak 20 data secara *realtime*. Pengambilan data tersebut setiap detik, sehingga jika 20 data maka membutuhkan waktu hanya 20 detik setiap pengukuran. Berikut merupakan kondisi pengujian *loss space* pada jarak 60 meter.



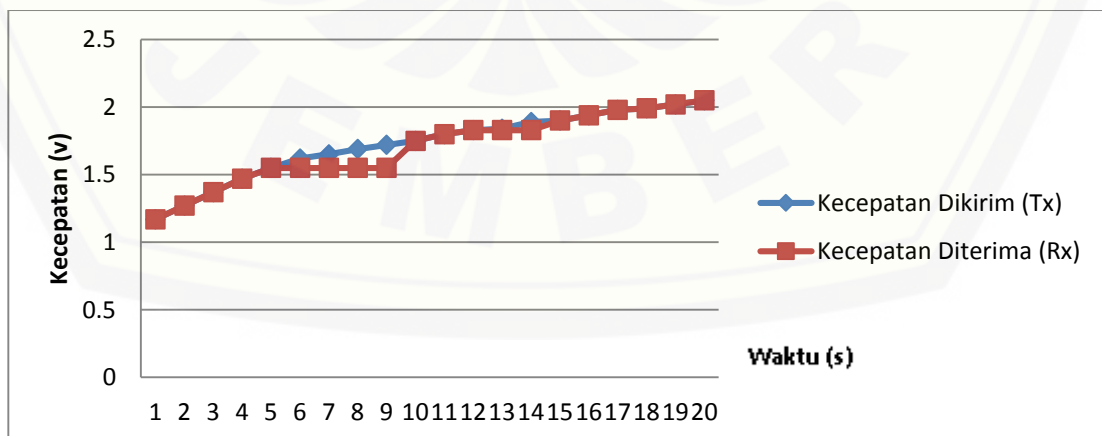
Gambar 4.17 Pengambilan data kondisi *loss space* pada 60 meter



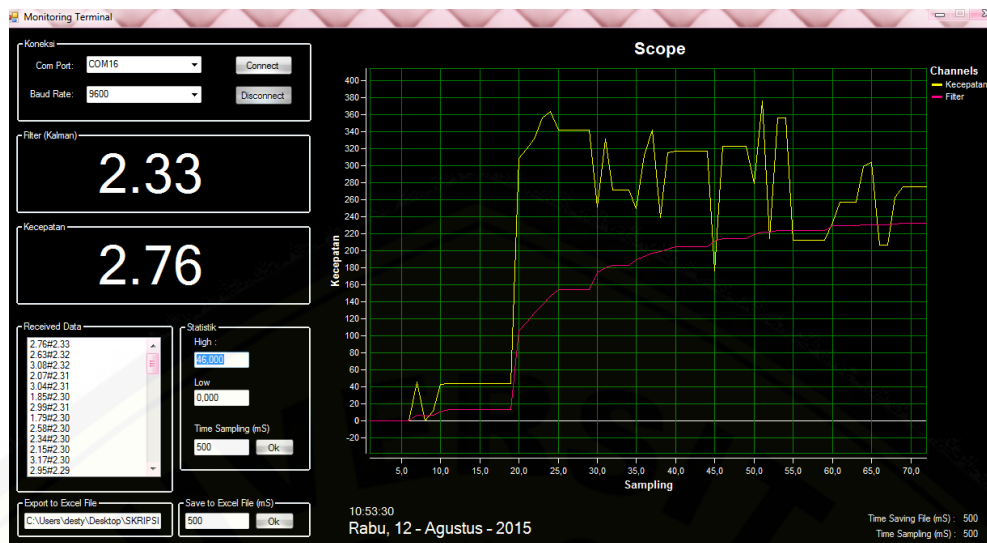
Tabel 4.3 merupakan data hasil pengujian *wireless* dalam kondisi *loss space* pada jarak 60 meter. Dapat dilihat grafiknya pada gambar 4.18 dan gambar 4.19 merupakan grafik perbandingan antara kecepatan yang dikirim dan diterima tanpa *Kalman Filter* dan grafik antara kecepatan dikirim dan yang diterima yang menggunakan *Kalman Filter*. Setelah diperoleh nilai *packet loss* masing- masing pengukuran kemudian dihitung nilai rata – rata *packet loss* dan diketahui nilai rata – rata *packet loss* keseluruhan untuk kecepatan tanpa *Kalman Filter* sebesar 0,076% dan rata – rata *packet loss* keseluruhan untuk kecepatan yang telah dilakukan *filter* menggunakan *Kalman Filter* sebesar 0,015%. Dapat dilihat pada tabel 4.3 bahwa nilai rata- rata *packet loss* lebih kecil dibandingkan pada pengukuran jarak 100 meter dan 80 meter. Hal ini berarti jarak juga dapat mempengaruhi proses transmisi.



Gambar 4.18 Grafik kecepatan angin dikirim (Tx) dan diterima (Rx)

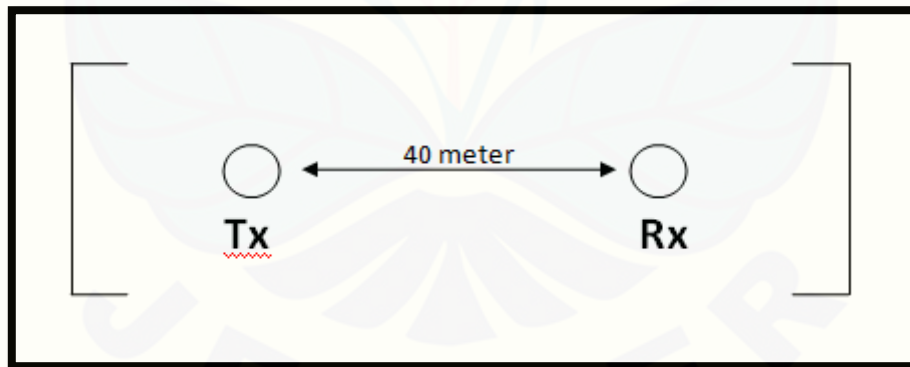


Gambar 4.19 Hasil *Kalman Filter* dikirim (Tx) dan diterima (Rx)



Gambar 4.20 Grafik kecepatan dan *Kalman Filter* pada jarak 60 meter

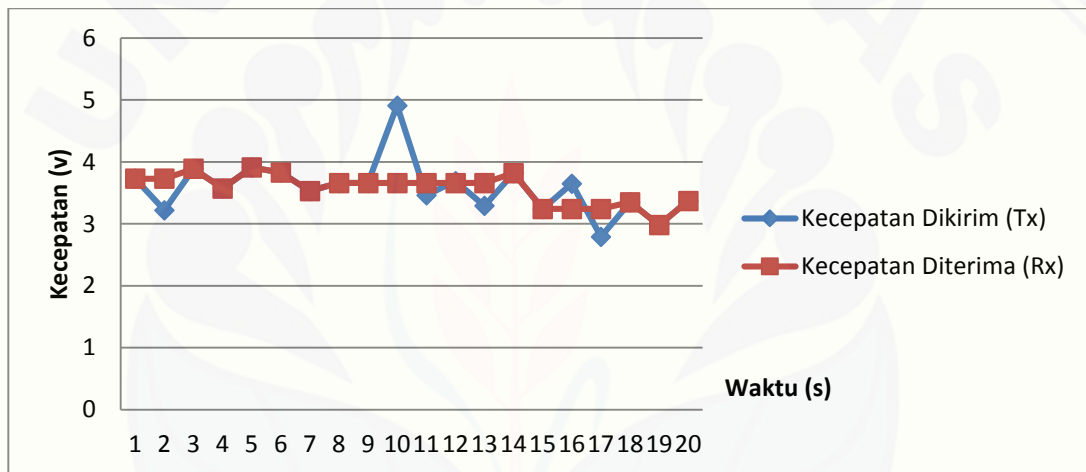
Pengambilan data keempat dengan jarak 40 meter dalam kondisi *loss space* dimana setiap pengukuran akan diambil sebanyak 20 data secara *realtime*. Pengambilan data tersebut setiap detik, sehingga jika 20 data maka membutuhkan waktu hanya 20 detik setiap pengukuran. Berikut merupakan kondisi pengujian *loss space* pada jarak 40 meter.



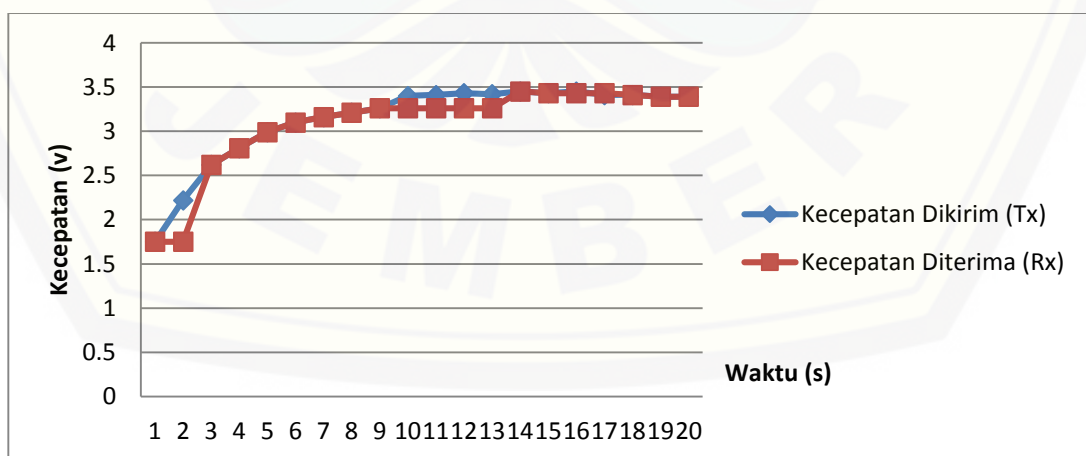
Gambar 4.21 Pengambilan data kondisi *loss space* pada jarak 40 meter



Tabel 4.4 merupakan data hasil pengujian *wireless* dalam kondisi *loss space* pada jarak 40 meter. Terlihat bahwa nilai *packet loss* yang lebih sedikit dan semakin kecil dibandingkan dengan pengukuran sebelumnya. Dapat dilihat grafiknya pada gambar 4.22 dan gambar 4.23 merupakan grafik perbandingan antara kecepatan yang dikirim dan diterima tanpa *Kalman Filter* dan grafik antara kecepatan dikirim dan yang diterima yang menggunakan *Kalman Filter*. Setelah diperoleh nilai *packet loss* masing - masing kemudian dihitung nilai rata – rata *packet loss* keseluruhan dan diketahui nilai rata – rata *packet loss* keseluruhan untuk kecepatan tanpa *Kalman Filter* sebesar 0,04% dan untuk rata – rata *packet loss* keseluruhan kecepatan yang menggunakan *Kalman Filter* sebesar 0,01%.



Gambar 4.22 Grafik kecepatan angin dikirim (Tx) dan diterima (Rx)

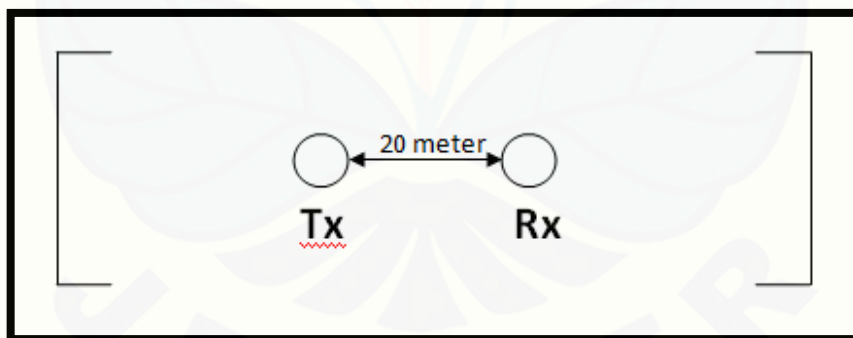


Gambar 4.23 Hasil *Kalman Filter* dikirim (Tx) dan diterima (Rx)



Gambar 4.24 Grafik kecepatan dan *Kalman Filter* pada jarak 40 meter

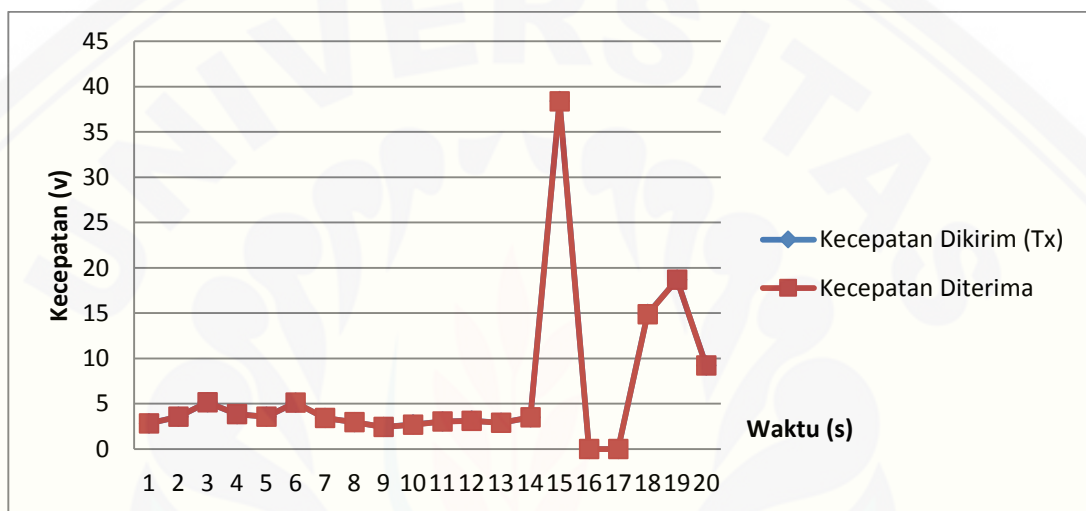
Pengambilan data kelima dengan jarak 20 meter dalam kondisi *loss space* dimana setiap pengukuran akan diambil sebanyak 20 data secara *realtime* yang dilakukan selama 20 detik. Berikut merupakan kondisi pengujian *loss space* pada jarak 20 meter.



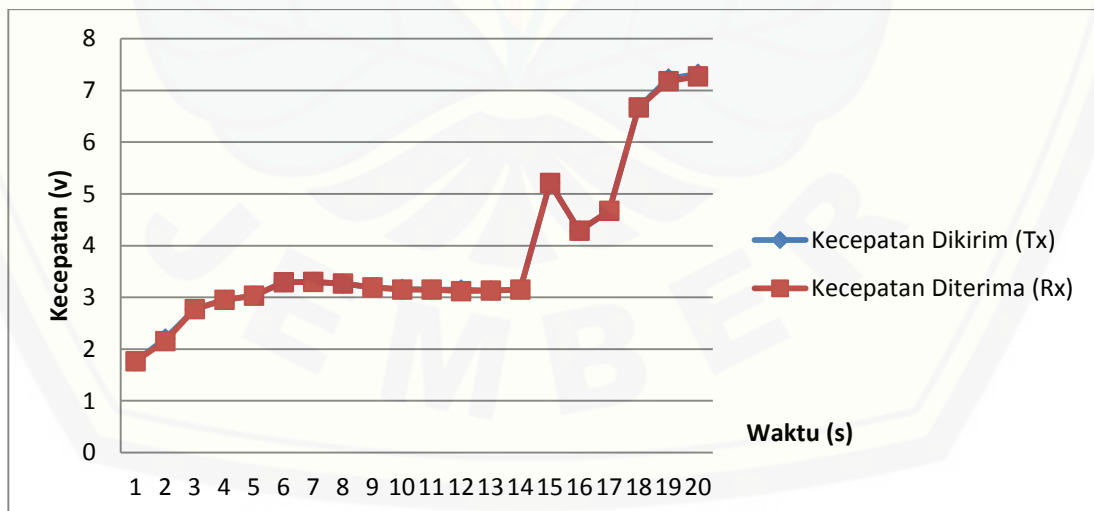
Gambar 4.25 Pengambilan data kondisi *loss space* pada jarak 20 meter



Tabel 4.5 merupakan data hasil pengujian *wireless* dalam kondisi *loss space* pada jarak 20 meter. Terlihat bahwa data yang diterima dari 20 data yang ada hampir 90% sama dengan data yang dikirimkan. Setelah diperoleh nilai *packet loss* masing- masing kemudian dihitung nilai rata – rata *packet loss* dan diketahui nilai rata – rata *packet loss* keseluruhan untuk kecepatan tanpa *Kalman Filter* sebesar 0,001% dan untuk rata – rata *packet loss* keseluruhan kecepatan yang menggunakan *Kalman Filter* sebesar 0,001%.



Gambar 4.26 Grafik kecepatan angin dikirim (Tx) dan diterima (Rx)



Gambar 4.27 Hasil *Kalman Filter* dikirim (Tx) dan diterima (Rx)



Gambar 4.28 Grafik kecepatan dan *Kalman Filter* pada jarak 20 meter

Berikut merupakan grafik dari *software Visual Basic* saat melakukan pengukuran. Dimana garis kuning merupakan kecepatan tanpa *Kalman Filter* dan untuk garis yang berwarna merah muda merupakan kecepatan menggunakan *Kalman Filter*. *Kalman Filter* akan selalu berada ditengah.

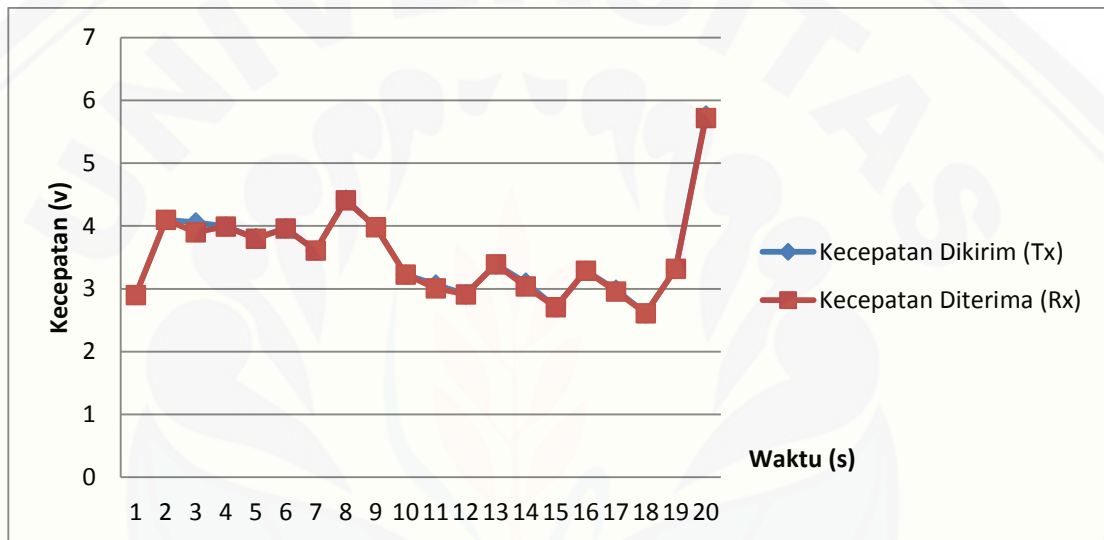
4.3.2 Pengujian *Wireless Kondisi Obstacle*

Pada pengujian kali ini dilakukan dalam kondisi ada halangan (*obstacle*) yang dilakukan antar rumah. Pada tahap pengambilan data dilakukan 3 kali pengambilan data dengan jarak yang berbeda yaitu 50 meter, 30 meter dan 20 meter. Tujuan pengujian pada kondisi ini ingin mengetahui apakah pengaruh terhadap data yang didapat dibandingkan dengan pada saat kondisi *loss space*. Serta menguji jarak maksimal yang bisa dijangkau oleh alat pengukur kecepatan angin tersebut dalam kondisi *obstacle*.

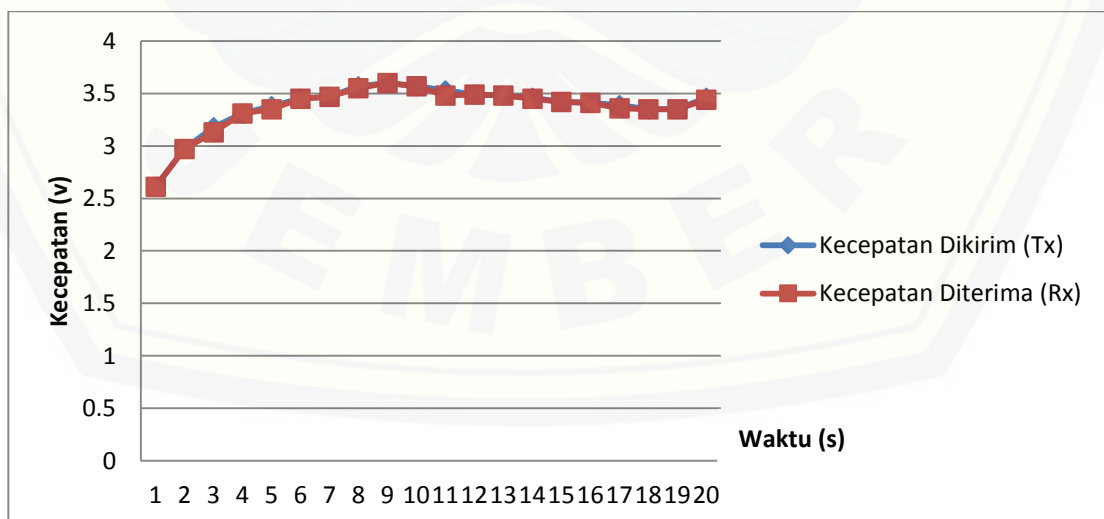
Pada pengambilan data pertama dilakukan pada jarak maksimal yaitu 30 meter dalam kondisi *obstacle* dimana setiap pengukuran akan diambil sebanyak 20 data secara *realtime*. Pengambilan data tersebut setiap detik, sehingga jika 20 data maka membutuhkan waktu hanya 20 detik setiap pengukuran. Berikut tabel hasil pengukurannya.



Tabel 4.6 merupakan data pengukuran pertama pada kondisi *obstacle* pada jarak 20 meter. Dapat dilihat pada tabel bahwa selisih antara data yang dikirim dan diterima cukup kecil, diketahui juga nilai *packet loss* pada masing – masing data. Setelah diperoleh nilai *packet loss* masing- masing kemudian dihitung nilai rata – rata *packet loss* keseluruhan dan diketahui nilai rata – rata *packet loss* keseluruhan untuk kecepatan tanpa *Kalman Filter* sebesar 0,005% dan rata – rata *packet loss* keseluruhan kecepatan yang menggunakan *Kalman Filter* sebesar 0,003%.



Gambar 4.29 Grafik kecepatan angin dikirim (Tx) dan diterima (Rx)



Gambar 4.30 Hasil *Kalman Filter* dikirim (Tx) dan diterima (Rx)



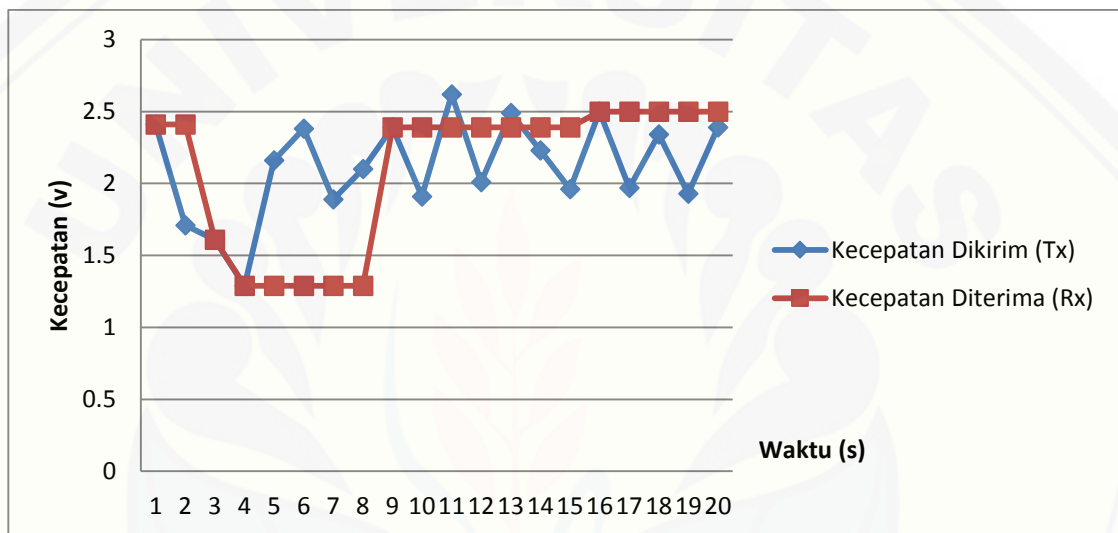
Gambar 4.31 Grafik kecepatan dan *Kalman Filter* pada jarak 20 meter

Disini *Kalman Filter* berfungsi untuk mengurangi *noise* agar tidak mengganggu pengiriman data. Dapat dilihat pada grafik diatas dimana sumbu yang berwarna kuning merupakan kecepatan tanpa *Kalman Filter* dan sumbu yang berwarna merah muda merupakan kecepatan menggunakan *Kalman Filter*. Apabila terjadi kenaikan atau penurunan nilai kecepatan angin yang drastis nilai *Kalman Filter* akan mengikuti kenaikan atau penurunannya secara perlahan. Dengan *Kalman Filter* ini data yang naik turun tadi dibuat stabil. Contoh pada data nomor 1 dengan kecepatan tanpa *Kalman Filter* sebesar 2,90 dan kecepatan dengan *Kalman Filter* sebesar 2,61, ketika kecepatan tanpa *Kalman Filter* naik menjadi 4,10 dan nilai kecepatan *Kalman Filter* 2,61 terbukti bahwa *Kalman Filter* tidak akan mengikuti kenaikan yang drastis tersebut.

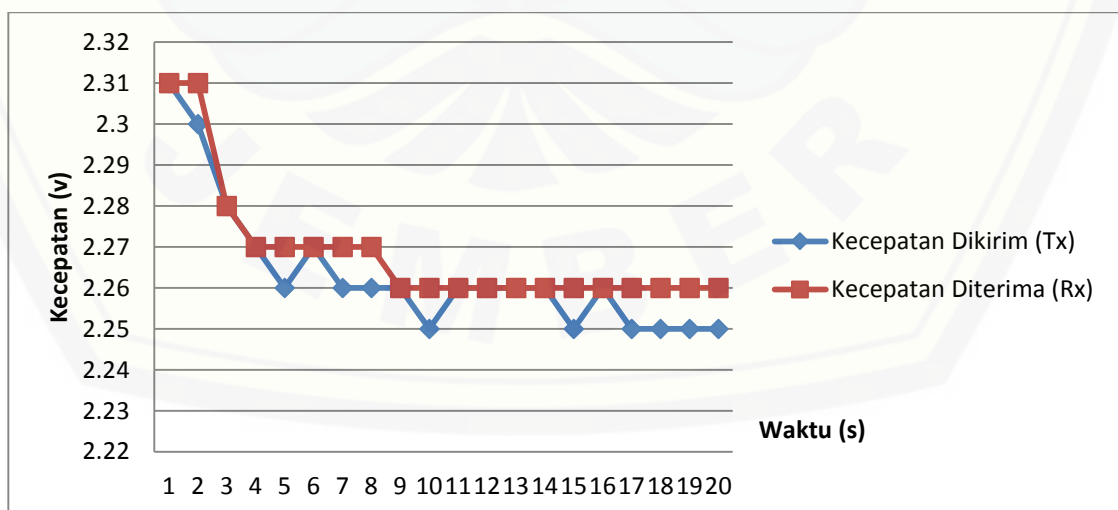
Pada pengambilan data kedua dilakukan pada jarak 30 meter dalam kondisi *obstacle* dimana setiap pengukuran akan diambil sebanyak 20 data secara *realtime*. Pengambilan data tersebut setiap detik, sehingga jika 20 data maka membutuhkan waktu hanya 20 detik setiap pengukuran.



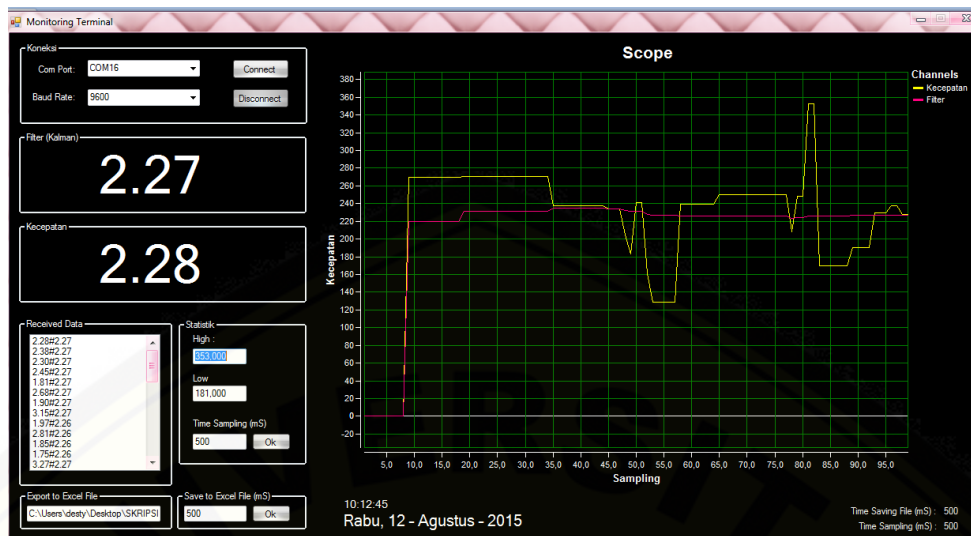
Tabel 4.7 merupakan data pengukuran kedua pada kondisi *obstacle* pada jarak 30 meter. Dapat dilihat pada tabel bahwa selisih antara data yang dikirim dan diterima cukup kecil, diketahui juga nilai *packet loss* pada masing – masing data yang semakin besar dibandingkan dengan pada pengukuran jarak 20 meter. Setelah diperoleh nilai *packet loss* masing- masing kemudian dihitung nilai rata – rata *packet loss* dan diketahui nilai rata – rata *packet loss* secara keseluruhan, untuk kecepatan tanpa *Kalman Filter* sebesar 0,171% dan untuk rata – rata *packet loss* keseluruhan kecepatan yang menggunakan *Kalman Filter* sebesar 0,002%.



Gambar 4.32 Grafik kecepatan angin dikirim (Tx) dan diterima (Rx)



Gambar 4.33 Hasil *Kalman Filter* dikirim (Tx) dan diterima (Rx)



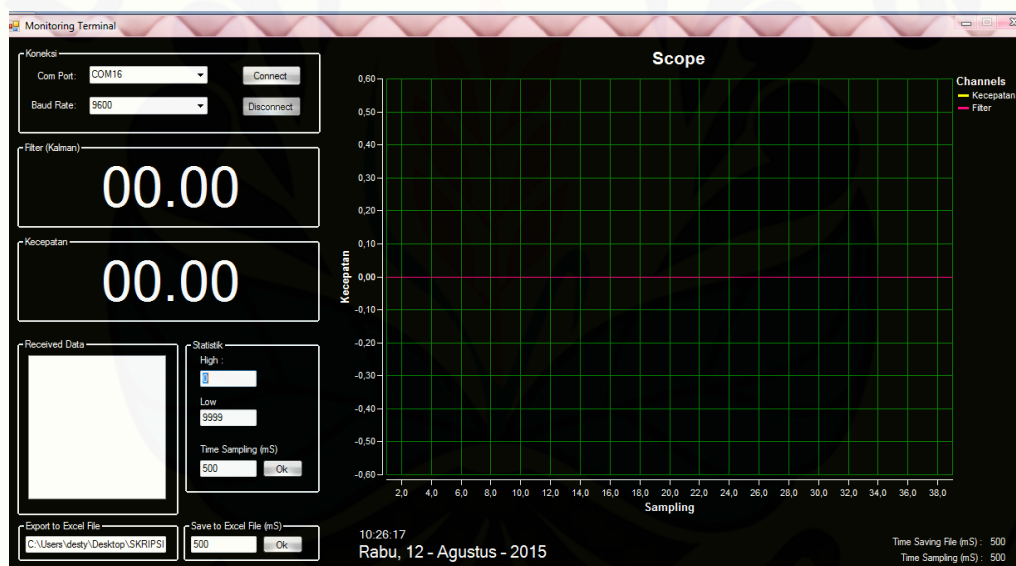
Gambar 4.34 Grafik kecepatan dan *Kalman Filter* pada jarak 30 meter

Dapat dilihat pada grafik di atas dimana sumbu yang berwarna kuning merupakan kecepatan tanpa *Kalman Filter* dan sumbu yang berwarna merah muda merupakan kecepatan menggunakan *Kalman Filter*. Apabila terjadi kenaikan atau penurunan nilai kecepatan angin yang drastis nilai *Kalman Filter* akan mengikuti kenaikan atau penurunannya secara perlahan. Dengan *Kalman Filter* ini data yang naik turun tadi dibuat stabil. Contoh pada data nomor 11 sampai dengan 14 dengan kecepatan tanpa *Kalman Filter* sebesar berkisar 2,6 dan kecepatan dengan *Kalman Filter* berkisar 2,26, ketika kecepatan tanpa *Kalman Filter* turun menjadi 1,96 dan nilai kecepatan *Kalman Filter* 2,5 terbukti bahwa *Kalman Filter* tidak akan mengikuti penurunan secara yang drastis namun hanya 0,1 saja penurunannya, sedangkan penurunan dari kecepatan tanpa *Kalman Filter* melebihi 1.

Pada pengambilan data ketiga dilakukan pada jarak 50 meter dalam kondisi *obstacle* dimana setiap pengukuran akan diambil sebanyak 20 data secara *realtime*. Pengambilan data tersebut setiap detik, sehingga jika 20 data maka membutuhkan waktu hanya 20 detik setiap pengukuran.



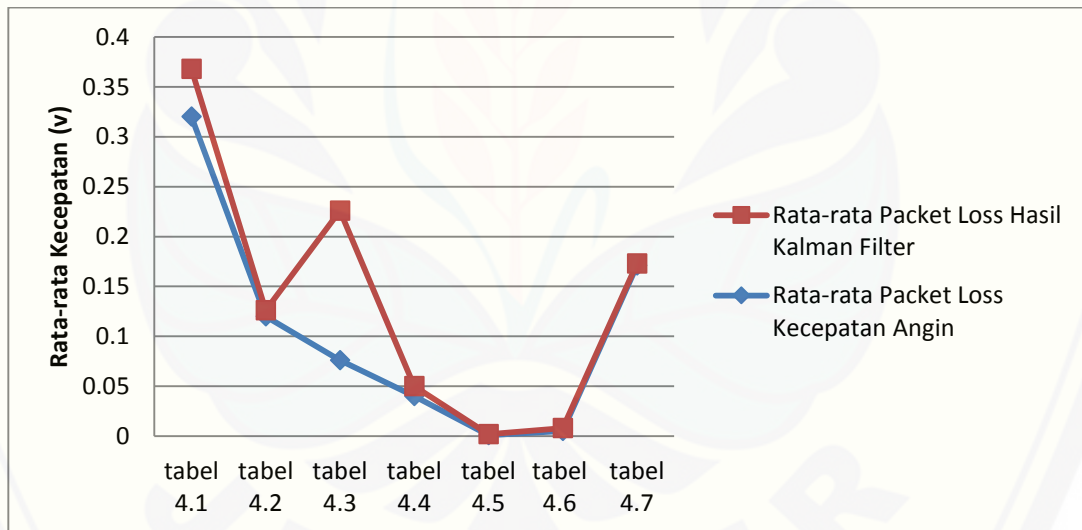
Tabel 4.8 merupakan data pengujian *wireless* pada kondisi *obstacle* pada jarak 50 meter. Terlihat data yang diterima baik data kecepatan angin yang tanpa *Kalman Filter* dan data kecepatan yang menggunakan *Kalman Filter* bernilai 0. Setelah melakukan pengujian ini dapat disimpulkan bahwa pada kondisi *obstacle* alat pengukur kecepatan angin ini mampu mengirimkan data dengan baik hanya sampai jarak 30 meter. Ketika mengambil data pada jarak 50 meter dengan halangan beberapa ruangan alat ini sudah tidak mampu mengirimkan data dengan baik atau data yang diterima bernilai 0 pada sisi penerima (Rx). Sehingga jarak maksimal pengukuran dalam kondisi *obstacle* yaitu hanya sampai 30 meter. Karena nilai kecepatan yang diterima bernilai 0 maka pada grafik antara garis yang menunjukkan kecepatan dan *Kalman Filter* akan lurus pada nilai 0 seperti pada gambar 4.35.



Gambar 4.35 Grafik kecepatan dan *Kalman Filter* pada jarak 50 meter

Tabel 4.9 Perbandingan Rata-rata *Packet Loss* Kecepatan Angin dengan Rata-rata *Packet Loss* Hasil *Kalman Filter* keseluruhan

Tabel	Rata-rata <i>Packet Loss</i> Kecepatan Angin	Rata-rata <i>Packet Loss</i> Hasil <i>Kalman Filter</i>
Tabel 4.1	0,32	0,048
Tabel 4.2	0,12	0,06
Tabel 4.3	0,076	0,015
Tabel 4.4	0,04	0,01
Tabel 4.5	0,001	0,001
Tabel 4.6	0,005	0,003
Tabel 4.7	0,171	0,002
Tabel 4.8	100	100



Gambar 4.36 Grafik Rata-rata *Packet Loss* Hasil *Kalman Filter* dan Kecepatan Angin

Pada grafik secara keseluruhan gambar 4.36 di atas, terlihat bahwa untuk data kondisi *loss space* berada pada tabel 4.1 sampai dengan tabel 4.5. Untuk tabel 4.1 sampai dengan tabel 4.5 urut mulai jarak 100 meter 80 meter 60 meter 40 meter dan 20 meter. Terlihat pada grafik terjadi penurunan, yang artinya semakin dekat jarak pengukuran nilai rata – rata *packet loss* semakin kecil. Sama halnya dengan

kondisi *obstacle* yaitu pada tabel 4.6 dan tabel 4.7, dimana tabel 4.6 merupakan jarak 20 meter dan tabel 4.7 merupakan jarak 30 meter. Terlihat terjadi kenaikan pada grafik, yang artinya semakin jauh atau semakin besar jarak saat pengukuran nilai *packet loss* semakin besar.

Dari kedua pengujian *wireless* yaitu dalam kondisi *loss space* dan *obstacle* terlihat perbandingannya bahwa proses pengiriman data pada saat kondisi *loss space* alat mampu menjangkau jarak yang lebih jauh yaitu 100 meter dibandingkan dalam kondisi *obstacle* yaitu hanya mampu sampai 30 meter. Hal ini berarti adanya penghalang mampu mempengaruhi proses pengiriman data yang dilakukan menggunakan sistem telemetri. Pada dasarnya *Kalman Filter* adalah *filter* sinyal masukkan dari *noise* yang diakibatkan ke tidak presisian pembacaan pada sensor, sehingga menyebabkan data kadang naik, kadang turun tiba - tiba dan kadang stabil. Hal inilah yang kadang menyebabkan suatu sistem pemrosesan sinyal kebingungan. Dengan *Kalman Filter* ini data yang naik turun tadi dibuat stabil. Jadi ketika data didapat oleh *Kalman*, data tersebut pasti akan berselisih dengan data aslinya (ini karena efek *Kalman*), namun ketika mendapatkan *noise*, *Kalman* tidak langsung memberi respon, tetapi menunggu beberapa saat sampai data benar- benar menunjukkan nilai beberapa iterasi (deretan data) yang masuk ke sistem.

Tabel 4.1 Pengujian *wireless* pada kondisi *loss space* jarak 100 meter

No	Kecepatan Angin		Packet Loss (%)	Hasil Kalman		Packet Loss (%)
	Dikirim (Tx)	Diterima (Rx)		Dikirim (Tx)	Diterima (Rx)	
1.	0,81	0,81	0	0,38	0,38	0
2.	0,53	0,53	0	0,43	0,43	0
3.	0,96	0,96	0	0,56	0,56	0
4.	0,98	0,98	0	0,64	0,64	0
5.	0,90	0,90	0	0,68	0,68	0
6.	1,32	1,32	0	0,77	0,77	0
7.	1,34	1,34	0	0,84	0,84	0
8.	1,92	1,92	0	0,96	0,96	0
9.	1,03	1,03	0	0,96	0,96	0
10.	1,47	1,03	0,29	1,01	0,96	0,04
11.	0,84	1,03	0,22	1,00	0,96	0,04
12.	0,87	1,03	0,18	0,99	0,96	0,03
13.	0,99	1,03	0,04	0,99	0,96	0,03
14.	19,17	0,99	0,94	2,19	0,99	0,54
15.	11,50	11,5	0	2,76	2,76	0
16.	12,02	11,5	0,04	3,31	2,76	0,16
17.	3,15	11,5	2,65	3,15	2,76	0,12
18.	3,03	0,68	0,77	3,03	3,03	0
19.	2,92	0,84	0,71	2,92	2,92	0
20.	2,82	0,92	0,67	2,82	2,82	0
Rata – rata packet loss (%)			0,32	Rata – rata packet loss (%)		0,048

Tabel 4.2 Pengujian *wireless* pada kondisi *loss space* jarak 80 meter

No	Kecepatan Angin		Packet Loss (%)	Hasil Kalman		Packet Loss (%)
	Dikirim (Tx)	Diterima (Rx)		Dikirim (Tx)	Diterima (Rx)	
1.	0,42	0,42	0	0,55	0,55	0
2.	0,66	0,66	0	0,57	0,57	0
3.	0,60	0,60	0	0,58	0,58	0
4.	0,91	0,91	0	0,63	0,63	0
5.	0,99	0,99	0	0,67	0,67	0
6.	0,80	0,80	0	0,68	0,68	0
7.	0,83	0,83	0	0,70	0,70	0
8.	1,24	0,83	0,33	0,75	0,70	0,06
9.	1,67	0,83	0,50	0,82	0,70	0,14
10.	1,18	0,83	0,29	0,85	0,70	0,17
11.	2,04	0,83	0,59	0,93	0,70	0,24
12.	2,34	2,34	0	1,03	1,03	0
13.	2,73	2,34	0,14	1,13	1,03	0,08
14.	2,83	2,34	0,17	1,23	1,03	0,16
15.	2,36	2,34	0,008	1,35	1,03	0,23
16.	4,25	2,34	0,44	1,50	1,03	0,31
17.	6,91	6,91	0	1,77	1,77	0
18.	1,19	1,19	0	1,74	1,74	0
19.	0,84	0,84	0	1,70	1,70	0
20.	0,68	0,68	0	1,66	1,66	0
Rata – rata packet loss (%)			0,12	Rata – rata packet loss (%)		0,06

Tabel 4.3 Pengujian *wireless* pada kondisi *loss space* jarak 60 meter

No	Kecepatan Angin		Packet Loss (%)	Hasil Kalman		Packet Loss (%)
	Dikirim (Tx)	Diterima (Rx)		Dikirim (Tx)	Diterima (Rx)	
1.	3,20	3,20	0	1,17	1,17	0
2.	3,32	3,32	0	1,27	1,27	0
3.	3,57	3,57	0	1,37	1,37	0
4.	3,64	3,64	0	1,47	1,47	0
5.	3,42	3,42	0	1,55	1,55	0
6.	3,39	3,42	0,008	1,62	1,55	0,04
7.	2,34	3,42	0,46	1,65	1,55	0,06
8.	2,54	3,42	0,34	1,69	1,55	0,08
9.	2,65	3,42	0,29	1,72	1,55	0,09
10.	2,52	2,52	0	1,75	1,75	0
11.	3,32	3,32	0	1,80	1,8	0
12.	2,72	2,72	0	1,83	1,83	0
13.	2,18	2,72	0,24	1,84	1,83	0,005
14.	3,38	2,72	0,19	1,89	1,83	0,03
15.	2,50	2,5	0	1,90	1,9	0
16.	3,13	3,13	0	1,94	1,94	0
17.	3,42	3,42	0	1,98	1,98	0
18.	2,39	2,39	0	1,99	1,99	0
19.	3,16	3,16	0	2,02	2,02	0
20.	3,18	3,18	0	2,05	2,05	0
Rata – rata packet loss (%)			0,076	Rata – rata packet loss (%)		0,015

Tabel 4.4 Pengujian *wireless* pada kondisi *loss space* jarak 40 meter

No	Kecepatan Angin		Packet Loss (%)	Hasil Kalman		Packet Loss (%)
	Dikirim (Tx)	Diterima (Rx)		Dikirim (Tx)	Diterima (Rx)	
1.	3,73	3,73	0	1,75	1,75	0
2.	3,22	3,73	0,15	2,22	1,75	0,17
3.	3,89	3,89	0	2,62	2,62	0
4.	3,57	3,57	0	2,81	2,81	0
5.	3,91	3,91	0	2,99	2,99	0
6.	3,83	3,83	0	3,10	3,1	0
7.	3,53	3,53	0	3,16	3,16	0
8.	3,66	3,66	0	3,21	3,21	0
9.	3,66	3,66	0	3,26	3,26	0
10.	4,91	3,66	0,25	3,40	3,26	0,04
11.	3,46	3,66	0,05	3,41	3,26	0,04
12.	3,69	3,66	0,008	3,43	3,26	0,04
13.	3,29	3,66	0,11	3,42	3,26	0,04
14.	3,82	3,82	0	3,45	3,45	0
15.	3,24	3,24	0	3,43	3,43	0
16.	3,65	3,24	0,11	3,45	3,43	0,005
17.	2,79	3,24	0,16	3,41	3,43	0,005
18.	3,35	3,35	0	3,41	3,41	0
19.	2,98	2,98	0	3,39	3,39	0
20.	3,37	3,37	0	3,39	3,39	0
Rata – rata packet loss (%)			0,04	Rata – rata packet loss (%)		0,01

Tabel 4.5 Pengujian *wireless* pada kondisi *loss space* jarak 20 meter

No	Kecepatan Angin		Packet Loss (%)	Hasil Kalman		Packet Loss (%)
	Dikirim (Tx)	Diterima (Rx)		Dikirim (Tx)	Diterima (Rx)	
1.	2.82	2.82	0	1.76	1.76	0
2.	3.57	3.55	0.005602	2.2	2.18	0.009
3.	5.14	5.14	0	2.77	2.77	0
4.	3.85	3.85	0	2.95	2.95	0
5.	3.57	3.57	0	3.03	3.03	0
6.	5.11	5.11	0	3.29	3.29	0
7.	3.43	3.43	0	3.3	3.3	0
8.	2.97	2.97	0	3.27	3.27	0
9.	2.42	2.42	0	3.19	3.19	0
10.	2.72	2.7	0.007353	3.16	3.15	0.003
11.	3.05	3.05	0	3.15	3.15	0
12.	3.14	3.11	0.009554	3.15	3.13	0.006
13.	2.9	2.9	0	3.13	3.13	0
14.	3.51	3.51	0	3.15	3.15	0
15.	38.39	38.35	0.001042	5.21	5.21	0
16.	0	0	0	4.29	4.29	0
17.	0	0	0	4.67	4.67	0
18.	14.91	14.89	0.001341	6.67	6.67	0
19.	18.69	18.69	0	7.23	7.2	0.004
20.	9.22	9.22	0	7.32	7.3	0.002
Rata – rata packet loss (%)			0,001	Rata – rata packet loss (%)		0,001

Tabel 4.6 Pengujian *wireless* pada kondisi *obstacle* jarak 20 meter

No	Kecepatan Angin		Packet Loss (%)	Hasil Kalman		Packet Loss (%)
	Dikirim (Tx)	Diterima (Rx)		Dikirim (Tx)	Diterima (Rx)	
1.	2,90	2,90	0	2,61	2,61	0
2.	4,10	4,10	0	2,97	2,97	0
3.	4,06	3,90	0,03	3,18	3,13	0,01
4.	3,99	3,99	0	3,31	3,31	0
5.	3,81	3,80	0,002	3,38	3,35	0,008
6.	3,96	3,96	0	3,45	3,45	0
7.	3,61	3,61	0	3,47	3,47	0
8.	4,42	4,41	0,002	3,57	3,55	0,005
9.	3,98	3,98	0	3,60	3,60	0
10.	3,23	3,23	0	3,57	3,57	0
11.	3,07	3,01	0,01	3,53	3,48	0,01
12.	2,91	2,91	0	3,49	3,49	0
13.	3,39	3,39	0	3,48	3,48	0
14.	3,10	3,04	0,01	3,46	3,45	0,002
15.	2,71	2,71	0	3,42	3,42	0
16.	3,29	3,29	0	3,41	3,41	0
17.	2,99	2,96	0,01	3,39	3,36	0,008
18.	2,61	2,61	0	3,35	3,35	0
19.	3,32	3,32	0	3,35	3,35	0
20.	5,76	5,72	0,006	3,46	3,44	0,005
Rata – rata packet loss (%)			0,005	Rata – rata packet loss (%)		0,003

Tabel 4.7 Pengujian *wireless* pada kondisi *obstacle* jarak 30 meter

No	Kecepatan Angin		Packet Loss (%)	Hasil Kalman		Packet Loss (%)
	Dikirim (Tx)	Diterima (Rx)		Dikirim (Tx)	Diterima (Rx)	
1.	2,41	2,41	0	2,31	2,31	0
2.	1,71	2,41	0,40	2,30	2,31	0,004
3.	1,61	1,61	0	2,28	2,28	0
4.	1,29	1,29	0	2,27	2,27	0
5.	2,16	1,29	0,40	2,26	2,27	0,004
6.	2,38	1,29	0,45	2,27	2,27	0
7.	1,89	1,29	0,31	2,26	2,27	0,004
8.	2,10	1,29	0,38	2,26	2,27	0,004
9.	2,39	2,39	0	2,26	2,26	0
10.	1,91	2,39	0,25	2,25	2,26	0,004
11.	2,62	2,39	0,08	2,26	2,26	0
12.	2,01	2,39	0,18	2,26	2,26	0
13.	2,49	2,39	0,04	2,26	2,26	0
14.	2,23	2,39	0,07	2,26	2,26	0
15.	1,96	2,39	0,21	2,25	2,26	0,004
16.	2,50	2,5	0	2,26	2,26	0
17.	1,97	2,5	0,26	2,25	2,26	0,004
18.	2,34	2,5	0,06	2,25	2,26	0,004
19.	1,93	2,5	0,29	2,25	2,26	0,004
20.	2,39	2,5	0,04	2,25	2,26	0,004
Rata – rata packet loss (%)			0,171	Rata – rata packet loss (%)		0,002

Tabel 4.8 Pengujian *wireless* pada kondisi *obstacle* jarak 50 meter

No	Kecepatan Angin		Packet Loss (%)	Hasil Kalman		Packet Loss (%)
	Dikirim (Tx)	Diterima (Rx)		Dikirim (Tx)	Diterima (Rx)	
1.	5,73	0	100	2,68	0	100
2.	6,93	0	100	4,04	0	100
3.	6,53	0	100	4,64	0	100
4.	9,53	0	100	5,59	0	100
5.	2,21	0	100	5,04	0	100
6.	5,74	0	100	5,10	0	100
7.	5,55	0	100	7,03	0	100
8.	9,03	0	100	7,25	0	100
9.	2,18	0	100	6,75	0	100
10.	5,41	0	100	6,63	0	100
11.	2,22	0	100	6,27	0	100
12.	6,41	0	100	6,28	0	100
13.	2,13	0	100	5,99	0	100
14.	4,69	0	100	5,90	0	100
15.	2,78	0	100	5,71	0	100
16.	2,84	0	100	5,54	0	100
17.	2,23	0	100	5,36	0	100
18.	5,57	0	100	5,37	0	100
19.	2,91	0	100	5,25	0	100
20.	3,15	0	100	5,15	0	100
Rata – rata packet loss (%)			100	Rata – rata packet loss (%)		100

BAB 5. PENUTUP

5.1 Kesimpulan

Setelah melakukan perencanaan dan pembuatan sistem kemudian dilakukan pengujian dan analisa, dari hasil tersebut dapat diambil beberapa kesimpulan sebagai berikut:

1. Perancangan *Kalman Filter* yang telah diterapkan pada alat pengukur kecepatan angin dapat dikatakan masih belum dapat bekerja dengan maksimal dalam proses pengiriman data, karena nilai hasil *Kalman Filter* yang diperoleh belum bisa dijadikan sebagai nilai yang mendekati dengan kecepatan angin yang asli atau tanpa *Kalman Filter* terbukti pada data tabel 4.1 sampai dengan tabel 4.7, padahal jika sesuai dengan teori bahwa seharusnya nilai kecepatan yang telah dilakukan *Kalman Filter* harus mendekati dengan nilai kecepatan angin yang asli atau yang tersimpan di *data logger*, sehingga jika dibandingkan antara nilai kecepatan yang telah dilakukan *Kalman Filter* dan nilai kecepatan angin yang asli nilai *packet loss* akan besar. *Kalman Filter* akan bekerja secara maksimal apabila data *offline*.
2. Analisa *packet loss* dari kedua kondisi saat pengujian sangat berbeda. Pada saat kondisi *loss space* alat mampu menjangkau jarak hingga 100 meter, sedangkan untuk kondisi *obstacle* alat hanya mampu menjangkau 30 meter saja. Untuk kondisi *loss space* dilakukan lima kali pengukuran dengan jarak yang berbeda-beda, dimana pada kondisi *loss space* pada jarak 20 meter memiliki nilai rata-rata *packet loss* yang paling kecil yaitu sebesar 0,001% (tabel 4.5, halaman 51) untuk kecepatan tanpa *Kalman Filter*, dan 0,001% (tabel 4.5, halaman 51) untuk kecepatan setelah dilakukan *Kalman Filter*. Dapat disimpulkan bahwa semakin dekat jaraknya maka nilai *packet loss* akan semakin kecil. Begitu pula untuk kondisi *obstacle*.

5.2 Saran

Setelah melakukan penelitian mengenai analisis *packet loss* pada alat pengukur kecepatan angin berbasis X-Bee Pro menggunakan *Kalman Filter* ini terdapat beberapa kekurangan atau kendala berikut ini merupakan saran untuk pengembangan lebih lanjut:

1. Diperlukan perangkat X-Bee Pro yang mempunyai daya pancar lebih jauh agar bisa melakukan pengukuran dari jarak yang lebih jauh dari penelitian ini.
2. Menggunakan *filter digital* yang lebih baik lagi supaya keutuhan data yang dikirimkan tetap terjaga.
3. Tidak disarankan menggunakan *Kalman Filter* untuk penelitian selanjutnya.
4. Agar nilai SNR (*Signal to Noise Rasio*) bagus, maka perlu dilakukan penyandian dan kendali *noise* menggunakan sandi – sandi koreksi antara lain Sandi *Hamming*, Sandi - sandi Siklis, Sandi - sandi BHC, Sandi - sandi *Reed Solomon* dan *Interleaving Blok*.

DAFTAR PUSTAKA

- Arriska, Afdhol. 2012. “*Rancangan dan Uji Coba Otomatisasi Irigasi Kendi*”. Bogor
- Dharmawan. 2014. “ Analisa Potensi Tenaga Angin Dengan Metode *Weibull Analysis* di Pantai Puger Kabupaten Jember”. Teknik Elektro Fakultas Teknik Universitas Jember.
- Ferdinando, Hany. 2010. “Dasar-dasar Sinyal dan Sistem”. Yogyakarta. ANDI.
- Hayadi, Suprpto. 2013. “Estimasi Sinyal Gamelan Menggunakan *Kalman Filter* Untuk Transkripsi”. Jurusan Teknik Elektro FTI – ITS.
- Kurniawan, Drs. L., Hanafi, Dr. E., Apriliani. 2014. “Penerapan Metode *Kalman Filter* Dalam Perbaikan Hasil Prediksi Cuaca Dengan Metode ARIMA”. Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Institut Teknologi Sepuluh November.
- Marta, Yuwono. 2015. “Arduino Itu Mudah”. Jakarta. PT.Elex Media Komputindo.
- Mirawati, Yasin. 2013. “ Prediksi Curah Hujan di Kota Semarang Dengan Metode *Kalman Filter*”. Jurusan Statistika FSM UNDIP.
- Pribadi, Anata. 2011. “PC *Data Logger* Berbasis Telemetry”. Jurnal Kompetensi Teknik vol.3, No. 1 Jurusan Teknik Elektro Fakultas Teknik Universitas Negeri Semarang.
- Rahman, Fathur. 2014. “Rancang Alat Pendeteksi Banjir Menggunakan Sistem Telemetry Berbasis Wireless Xbee Pro”. Jurusan Teknik Elektro Fakultas Teknik Universitas Jember.
- Pambudi, Rizka Nugraha. 2012. “Sistem Telemetry Pemantau Gempa Menggunakan Jaringan GSM”. KK FTETI Prodi Fisika FMIPA Institut Teknologi Bandung.
- Septiadhi. 2011. “Pemanfaatan Modul *Transmitter-Receiver* Untuk Pengukuran Telemetry Unsur-Unsur Cuaca”. Fakultas Matematika dan Ilmu Pengetahuan Alam Institut Pertanian Bogor.
- Simbolon, Sani. 2014. “Analisis Pengurangan *Derau* Pada Sinyal *Loudspeaker* Menggunakan Filter Adaptif Kalman”. Konsentrasi Teknik Telekomunikasi Fakultas Teknik Universitas Sumatera Utara.

Supeno.dkk. 2012. "Disain *Wireless Funcional Electrical Stimulator* Menggunakan X-Bee Pro". Malang.

Susanto Heri. "Perancangan Sistem Telemetri *Wireless* Untuk Mengukur Suhu dan Kelembaban Berbasis Arduino Uno R3 ATMEGA328P dan Xbee Pro", Jurusan Teknik Elektro Fakultas Teknik Universitas Maritim Raja Ali Haji.

Triawan,Yudi. 2011. "Pemanfaatan Modul *Transmitter-Receiver* Untuk Pengukuran Telemetri Unsur-Unsur Cuaca", Fakultas Matematika Dan Ilmu Pengetahuan Alam Institut Pertanian Bogor.



LAMPIRAN

a. Listing program Arduino 1.6.5

```
/*
```

```
SD card test
```

This example shows how use the utility libraries on which the SD library is based in order to get info about your SD card.

Very useful for testing a card when you're not sure whether its working or not.

The circuit:

- * SD card attached to SPI bus as follows:

- ** MOSI - pin 11 on Arduino Uno/Duemilanove/Diecimila

- ** MISO - pin 12 on Arduino Uno/Duemilanove/Diecimila

- ** CLK - pin 13 on Arduino Uno/Duemilanove/Diecimila

- ** CS - depends on your SD card shield

or module.

```
*/
```

```
#include <FreqCount.h>
```

```
#include <SPI.h>
```

```
// include the SD library:
```

```
#include "LiquidCrystal.h";
```

```
LiquidCrystal lcd(12, 11, 5, 4, 3,2);
```

```
#include <SD.h>
```

```
// how many milliseconds between grabbing data and logging it. 1000 ms is  
once a second
```

```
#define LOG_INTERVAL 1000 // mills between entries (reduce to take  
more/faster data)
```

```
// how many milliseconds before writing the logged data permanently to disk
// set it to the LOG_INTERVAL to write each time (safest)
// set it to 10*LOG_INTERVAL to write all data every 10 datareads, you could
lose up to
// the last 10 reads if power is lost but it uses less power and is much faster!
#define SYNC_INTERVAL 1000 // mills between calls to flush() - to write
data to the card
uint32_t syncTime = 0; // time of last sync()
// the digital pins that connect to the LEDs
#define redLEDPin 1
#define buzzer 9

//float varKecepatan = 0.024537879;
//float varKecepatan = 0.058788479;
float varKecepatan = 1.136115;
float varProcess = 1e-8;
float Pc = 0.0;
float G = 0.0;
float P = 1.0;
float Xp = 0.0;
float Zp = 0.0;
float Xe = 0.0;

int Phi = 3.1428571428571428571428571428571;
int JariJari = 1.8;
float frekuensi;
// for the data logging shield, we use digital pin 10 for the SD cs line
const int chipSelect = 53;
// the logging file
File logfile;
void error(char *str)
```

```
{
  lcd.setCursor(0,0);
  lcd.print("PASANG MEMORI.! ");
  lcd.setCursor(1,1);
  lcd.print("LOG DATA GAGAL");
  Serial.println(str);
  // red LED indicates error
  digitalWrite(redLEDpin, HIGH);

  while(1);
}
void setup(void)
{
  lcd.begin(16, 2);
  Serial.begin(9600);
  //Serial.print("SIP");
  // Tulis Temperatur di LCD
  pinMode(redLEDpin, OUTPUT);
  pinMode(buzzer, OUTPUT);
  #if WAIT_TO_START
  Serial.println("Type any character to start");
  while (!Serial.available());
  #endif //WAIT_TO_START
  // initialize the SD card
  //Serial.print("Initializing SD card...");
  // make sure that the default chip select pin is set to
  // output, even if you don't use it:
  pinMode(53, OUTPUT);
  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    error("Card failed, or not present");
```

```
}  
//Serial.println("card initialized.");  
// create a new file  
char filename[] = "LOGGER00.CSV";  
for (uint8_t i = 0; i < 100; i++) {  
  filename[6] = i/10 + '0';  
  filename[7] = i%10 + '0';  
  if (! SD.exists(filename)) {  
    // only open a new file if it doesn't exist  
    logfile = SD.open(filename, FILE_WRITE);  
    break; // leave the loop!  
  }  
}  
if (! logfile) {  
  error("couldnt create file");  
}  
lcd.setCursor(0,0);  
lcd.print("FILE SUKSES: ");  
lcd.setCursor(1,1);  
lcd.println(filename);  
  delay(5000);  
  lcd.clear();  
logfile.println("Freq >> RPM >> Kecepatan angin");  
#if ECHO_TO_SERIAL  
  Serial.println("Freq >> RPM >> Kecepatan angin");  
#endif //ECHO_TO_SERIAL  
  
  lcd.print("Freq>>RPM>>Kec");  
  FreqCount.begin(1000);  
}  
void loop(void)
```



```
{
// delay for the amount of time we want between readings
delay((LOG_INTERVAL - 1) - (millis() % LOG_INTERVAL));
digitalWrite(buzzer, HIGH);
// fetch the time

//Pencacahan frekuensi
if (FreqCount.available())
{
unsigned long count = FreqCount.read();
lcd.setCursor(0, 1);
lcd.print(count / 32);
lcd.print(" ");
float RPM;
lcd.setCursor(5,1);
RPM = (((float)count * (60.0/70.0)) / 32) ;
lcd.print(RPM);
lcd.print(" ");

float KecepatanAngin;
float RPM2Rad;
RPM2Rad = ((2.0 * Phi/60.0) * RPM);
lcd.setCursor(11,1);
KecepatanAngin = RPM2Rad * JariJari;
lcd.print(KecepatanAngin);
lcd.print(" ");

Pc = P + varProcess;
G = Pc / (Pc + varKecepatan);
P = (1 - G) * Pc;
Xp = Xe;
```

```
Zp = Xp;  
Xe = G * (KecepatanAngin - Zp) + Xp;
```

```
Serial.print(KecepatanAngin);  
Serial.print("#");  
Serial.println(Xe);
```

```
logfile.print(" ");  
logfile.print(count);  
logfile.print(" ");  
logfile.print(RPM);  
logfile.print(" ");  
logfile.print(KecepatanAngin);  
logfile.print(" ");  
logfile.print(Xe);
```

```
#if ECHO_TO_SERIAL  
Serial.print(" ");  
Serial.print(count);  
#endif // ECHO_TO_SERIAL  
logfile.println();  
#if ECHO_TO_SERIAL  
Serial.println();  
#endif //ECHO_TO_SERIAL
```

```
// Now we write data to disk! Don't sync too often - requires 2048 bytes of I/O  
to SD card
```

```

// which uses a bunch of power and takes time
if ((millis() - syncTime) < SYNC_INTERVAL) return;
syncTime = millis();
// blink LED to show we are syncing data to the card & updating FAT!
digitalWrite(redLEDPin, HIGH);
logfile.flush();
digitalWrite(redLEDPin, LOW);
}
}

```

b. Listing Program Visual Basic

```

Imports System
Imports System.ComponentModel
Imports System.Threading
Imports System.IO.Ports
Imports Mitov.PlotLab
Imports Excel = Microsoft.Office.Interop.Excel

```

Public Class frmDisplay

```

    Dim comOpen As Boolean    'Keeps track of the port status. True = Open;
    False = Closed

```

```

    Dim readbuffer As String  'Buffer of whats read from the serial port

```

```

    Dim time As Integer = 0

```

```

    Dim plotScopeA As Double

```

```

    Dim plotScopeB As Double

```

```

    Dim plotAverage As Double

```

```

    Dim timeSet As Integer

```

```

    Dim timeSFE As Integer

```

```

    Dim APP As Excel.Application

```

```
Dim worksheet As Excel.Worksheet
```

```
Dim workbook As Excel.Workbook
```

```
Dim y As Integer = 2
```

```
Private Sub frmDisplay_Load(ByVal sender As Object, ByVal e As  
EventArgs) Handles MyBase.Load
```

```
OpenFileDialog1.Filter = "Excel (*.xlsx)|*.xlsx"
```

```
If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
```

```
    namafile.Text = OpenFileDialog1.FileName
```

```
End If
```

```
APP = New Excel.Application
```

```
workbook = APP.Workbooks.Open(namafile.Text)
```

```
worksheet = workbook.Worksheets("sheet1")
```

```
'Get all connected serial ports
```

```
Dim comPorts As String() = System.IO.Ports.SerialPort.GetPortNames
```

```
Tanggal.Text = Format(Now, "dddd, dd – MMMM – yyyy")
```

```
Jam.Text = TimeString
```

```
If comPorts.Count < 1 Then
```

```
    'If there are not ports connected, show an error and close the program.
```

```
    MsgBox("There are no com ports available! Closing program.")
```

```
    Me.Close()
```

```
Else
```

```
    cmbPort.Items.AddRange(comPorts)
```

```
    cmbPort.Text = comPorts(0)
```

```
End If
```

```
Scope1.XAxis.AxisLabel.Text = "Sampling"
```

```
Scope1.YAxis.AxisLabel.Text = "Kecepatan"
```

```
End Sub
```

```
Private Sub frmDisplay_FormClosing(ByVal sender As Object, ByVal e As  
FormClosingEventArgs) Handles Me.FormClosing
```

```
'Gracefully disconnect before form closes
```

```
DoDisconnect()
```

```
End Sub
```

```
Private Sub SerialPort1_DataReceived(ByVal sender As System.Object, _
```

```
ByVal e As System.IO.Ports.SerialDataReceivedEventArgs) _
```

```
Handles SerialPort1.DataReceived
```

```
    If comOpen Then
```

```
        Try
```

```
            'Send data to a new thread to update the ph display
```

```
            readbuffer = SerialPort1.ReadLine()
```

```
            Me.Invoke(New EventHandler(AddressOf updateTemp))
```

```
        Catch ex As Exception
```

```
            'Otherwise show error. Will display when disconnecting.
```

```
            'MsgBox(ex.Message)
```

```
        End Try
```

```
    End If
```

```
End Sub
```

```
Public Sub updateTemp(ByVal sender As Object, ByVal e As  
System.EventArgs)
```

```
    'Update ph display as it comes in
```

```
    Dim kode As Integer = 0
```

```
    Dim read As String
```

```
    Dim aryTextFile() As String
```

```
read = readbuffer.Replace(vbCr, "").Replace(vbLf, "")
LstHistory.Items.Insert(0, read)
aryTextFile = read.Split("#")
```

```
Kecepatan.Text = aryTextFile(0)
Kalman.Text = aryTextFile(1)
plotScopeA = aryTextFile(0)
plotScopeB = aryTextFile(1)
```

```
'Check Highest Temp
If TxtHigh.Text < aryTextFile(0) Then
    TxtHigh.Text = aryTextFile(0)
    TxtHigh.Text = FormatNumber(TxtHigh.Text, 3)
End If
```

```
'Check Lowest Temp
If TxtLow.Text > aryTextFile(0) Then
    TxtLow.Text = aryTextFile(0)
    TxtLow.Text = FormatNumber(TxtLow.Text, 3)
End If
```

```
End Sub
```

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
```

```
    time = time + 1
    Jam.Text = TimeString
    Tanggal.Text = Format(Now, "dddd, dd - MMMM - yyyy")
    Scope1.Channels(0).Data.AddXYPoint(time, plotScopeA)
    Scope1.Channels(1).Data.AddXYPoint(time, plotScopeB)
```

End Sub

Public Sub DoDisconnect()

'Graceful disconnect if port is open

If comOpen Then

SerialPort1.DiscardInBuffer()

SerialPort1.Close()

'Reset our flag and controls

comOpen = False

btnDisconnect.Enabled = False

btnConnect.Enabled = True

cmbBaud.Enabled = True

cmbPort.Enabled = True

My.Computer.Audio.Stop()

End If

End Sub

Public Sub DoConnect()

'Setup the serial port connection

With SerialPort1()

.PortName = cmbPort.Text 'Selected Port

.BaudRate = CInt(cmbBaud.Text) 'Baud Rate. 9600 is default.

.Parity = IO.Ports.Parity.None

.DataBits = 8

.StopBits = IO.Ports.StopBits.One

.Handshake = IO.Ports.Handshake.None

.RtsEnable = False

.ReceivedBytesThreshold = 1

.NewLine = vbCr

```
.ReadTimeout = 10000
```

```
End With
```

```
'Try to open the selected port...
```

```
Try
```

```
    SerialPort1.Open()
```

```
    comOpen = SerialPort1.IsOpen
```

```
Catch ex As Exception
```

```
    comOpen = False
```

```
    MsgBox("Error Open: " & ex.Message)
```

```
End Try
```

```
btnDisconnect.Enabled = True
```

```
btnConnect.Enabled = False
```

```
cmbBaud.Enabled = False
```

```
cmbPort.Enabled = False
```

```
End Sub
```

```
Private Sub btnConnect_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles btnConnect.Click
```

```
    'Conect to serial port
```

```
    DoConnect()
```

```
    Timer1.Enabled = True
```

```
    Timer2.Enabled = True
```

```
End Sub
```

```
Private Sub btnDisconnect_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles btnDisconnect.Click
```

```
    'Disconnect the serial port
```

```
    Timer1.Enabled = False
```

```
    Timer2.Enabled = False
```



```
workbook.Save()  
DoDisconnect()  
End Sub
```

```
Private Sub Ok_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles Ok.Click  
    t1.Text = Timer1.Interval  
    timeSet = TimeBox.Text  
    Timer1.Interval = timeSet  
End Sub
```

```
Private Sub Timer2_Tick(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles Timer2.Tick  
    worksheet.Cells(y, 1).Value = DateString  
    worksheet.Cells(y, 2).Value = TimeString  
    worksheet.Cells(y, 3).Value = Kecepatan.Text  
    worksheet.Cells(y, 4).Value = Kalman.Text  
    y = y + 1  
    workbook.Save()  
End Sub
```

```
Private Sub Oks_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles Oks.Click  
    timeSFE = SEF.Text  
    Timer2.Interval = timeSFE  
    t2.Text = Timer2.Interval  
End Sub
```

```
End Class
```

c. Dokumentasi Alat

