TUGAS AKHIR – EE 184801

LOKALISASI SUARA PADA *UNMANNED AERIAL VEHICLE* DENGAN METODE *INTERAURAL TIME DIFFERENCE* MENGGUNAKAN MIKROKONTROLER STM32F429

Yulizar Edo Pratama Cordova
NRP 07111540000125

Dosen Pembimbing
Astria Nur Irfansyah, ST., MT., Ph.D.

DEPATERMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman Ini Sengaja Dikosongkan

TUGAS AKHIR – EE 184801

LOKALISASI SUARA PADA *UNMANNED AERIAL VEHICLE* DENGAN METODE *INTERAURAL TIME DIFFERENCE* MENGGUNAKAN MIKROKONTROLER STM32F429

Yulizar Edo Pratama Cordova
NRP 07111540000125

Dosen Pembimbing
Astria Nur Irfansyah, ST., MT., Ph.D.

DEPATERMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman Ini Sengaja Dikosongkan

FINAL PROJECT – EE 184801

SOUND LOCALIZATION ON UNMANNED AERIAL VEHICLE WITH INTERAURAL TIME DIFFERENCE METHOD USING MICROCONTROLLER STM32F429

Yulizar Edo Pratama Cordova
NRP 07111540000125

Supervisor(s)
Astria Nur Irfansyah, ST., MT., Ph.D.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman Ini Sengaja Dikosongkan

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**LOKALISASI SUARA PADA *UNMANNED AERIAL VEHICLE* DENGAN METODE *INTERAURAL TIME DIFFERENCE* MENGGUNAKAN MIKROKONTROLER STM32F429**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Januari 2020

Yulizar Edo Pratama Cordova
NRP. 0711 15 4000 0125

Halaman Ini Sengaja Dikosongkan

LOKALISASI SUARA PADA *UNMANNED AERIAL VEHICLE* DENGAN METODE *INTERAURAL TIME DIFFERENCE* MENGGUNAKAN MIKROKONTROLER STM32F429

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I

Astrin Nur Hidayah, ST., MT., Ph.D.
NIP. 198103252010121002

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
DEPARTEMEN TEKNIK ELEKTRO

SURABAYA
JANUARI, 2020

Halaman Ini Sengaja Dikosongkan

LOKALISASI SUARA PADA *UNMANNED AERIAL VEHICLE* DENGAN METODE *INTERAURAL TIME DIFFERENCE* MENGGUNAKAN MIKROKONTROLER STM32F429

Nama : Yulizar Edo Pratama Cordova
Pembimbing : Astria Nur Irfansyah, ST., M.Eng., Ph.D

ABSTRAK

Dalam bidang SAR (*Search and Rescue*) banyak penelitian menggunakan UAV (*Unmanned Aerial Vehicle*) untuk mencari *survivor* atau orang yang sedang mempertahankan diri dalam hutan bisa karena tersesat atau penyebab lainnya. Pencarian banyak dilakukan dengan penginderaan secara visual dimana objek dapat ditangkap oleh perangkat keras yang tertanam pada UAV. Jika penjelejahan dilakukan pada malam hari dengan pencahayaan yang kurang, maka metode yang digunakan adalah dengan memanfaatkan sumber suara yang datang untuk menentukan lokasi survivor tersebut.

Indera pendengaran manusia dapat menentukan arah sumber suara dengan mengestimasi sumber suara datang dengan akurasi yang tinggi. Manusia dapat mengenali lokasi suara secara 3 dimensi dan dapat mengenali keberadaan sumber suara pada posisi secara 180 derajat dimana tanpa adanya pembacaan secara visual.

Penelitian kali ini penulis membuat alat untuk menentukan arah sumber suara dengan 2 mikrofon dan STM32F429. Metode *Interaural Time Difference* digunakan untuk menentukan arah suara dengan cara membandingkan perbedaan waktu datang dari sumber suara. Adanya UAV tipe *quadcopter* untuk membawa alat tersebut sehingga dapat dilakukan penjelajahan dan mempresentasikan hasil pembacaan secara *wireless* melalui saluran bluetooth. *Noise* yang dihasilkan motor dapat ditanggulangi dengan mikrofon MEMS yang memiliki *signal-to-noise ratio* yang bagus. Hasil pengujian peneletian ini ketika dalam ruangan dengan pembacaan 5 sudut memiliki rata rata *error* pembacaan sebesar 15 % dan untuk pembacaan 3 posisi dalam keadaan terbang lapangan sudah sesuai.

Kata Kunci: Arah Sumber Suara, *Interaural Time Difference,* UAV, Kebisingan, mikrofon MEMS

Halaman Ini Sengaja Dikosongkan

SOUND LOCALIZATION ON UNMANNED AERIAL VEHICLE WITH INTERAURAL TIME DIFFERENCE METHOD USING MICROCONTROLLER STM32F429

Name : Yulizar Edo Pratama Cordova
Supervisor : Astria Nur Irfansyah, ST., M.Eng., Ph.D

ABSTRACT

In the field of SAR (Search and Rescue) a lot of research using UAV (Unmanned Aerial Vehicle) to look for survivors or people who are defending themselves in the forest can be due to getting lost or other causes. Searching is mostly done by visual sensing where objects can be captured by hardware that is embedded in the UAV. If the exploration is done at night with poor lighting, the method used is to utilize the sound source that comes to determine the location of the survivor.

The human sense of hearing can determine the direction of the sound source by estimating the source of the sound coming with high accuracy. Humans can recognize the location of sound in 3 dimensions and can recognize the existence of sound sources in a position of 180 degrees where there is no visual reading.

This research the author makes a tool to determine the direction of the sound source with 2 microphones and STM32F429. The Interaural Time Difference method is used to determine the direction of the sound by comparing the time difference coming from the sound source. The existence of a quadcopter type UAV to carry the device so that it can be explored and presented the results of readings wirelessly via Bluetooth channels. The noise generated by the motor can be overcome with a MEMS microphone that has a good signal-to-noise ratio. The results of this study when in a room with a reading of 5 angles have an average reading error of 15% and for reading 3 positions in a field flight state are appropriate.

Keywords: Sound Source Direction, Interaural Time Difference, UAV, Noise, MEMS microphone

Halaman Ini Sengaja Dikosongkan

KATA PENGANTAR

Puji dan syukur kepada Allah SWT atas berkat dan hikmat yang diberikan, penulis dapat menyelesaikan laporan tugas akhir dengan judul "**LOKALISASI SUARA PADA *UNMANNED AERIAL VEHICLE* DENGAN METODE *INTERAURAL TIME DIFFERENCE* MENGGUNAKAN MIKROKONTROLER STM32F429**". Sebagai salah satu persyaratan dalam menyelesaikan Pendidikan program Strata-Satu Departemen Teknik Elektro, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember.

Pada kesempatan ini penulis ingin mengucapkan ucapan terimakasih yang sebesar- besarnya kepada beberapa pihak yang telah memberikan dukungan selama proses pengerjaan tugas akhir ini, antara lain:

1. Orang tua beserta keluarga penulis, yang tak henti-hentinya memberikan semangat, dukungan dan kasih sayang yang luar biasa kepada penulis.
2. Kepala Departemen Teknik Elektro ITS Dedet Candra Riawan, ST., M.Eng., Ph.D. atas izin dan kesempatan yang diberikan kepada penulis untuk melaksanakan tugas akhir ini.
3. Kepala Laboraturium Mikroelektronika dan Sistem Tertanam Dr. Ir. Hendra Kusuma, M.Eng.Sc. yang telah memberi fasilitas laboraturium yang lengkap untuk merancang alat tugas akhir penulis.
4. Bapak Astria Nur Irfansyah, ST., M.Eng., Ph.D selaku dosen pembimbing atas gagasan topik penelitian untuk tugas akhir serta bimbingan dan arahan untuk penulis selama mengerjakan tugas akhir.
5. Bapak Ir. Djoko Purwanto, M.Eng.,Ph.D., Bapak Ronny Mardiyanto, ST.,MT.,Ph.D., Bapak Ir. Harris Pirngadi, MT., Bapak Fajar Budiman, ST.,M.Sc. selaku dosen penguji yang telah memberikan kritik dan saran mengenai tugas akhir serta memberi masukan.
6. Muh Alif Farabi Eranugroho, ST. yang telah membantu pengerjaan tugas akhir saya dibidang pemrosesan sinyal digital, pemrograman dan pembelajaran mengenai teori-teori dalam pemrosesan sinyal digital sehingga tugas akhir saya dapat berhasil dengan baik.

7. Satria Bhaskara Adinugraha yang telah membantu pengerjaan tugas akhir saya dengan pengenalan modul STM32F429 serta cara menggunakannya.
8. Hamid Yusuf Putranto yang telah membantu pengerjaan tugas akhir saya dibidang UAV mengenai pengelanan komponen *Quadcopter* serta cara merakit dan menggunakannya.
9. Rama Kharisma yang telah membantu mencari lokasi toko di Singapura untuk komponen penting tugas akhir saya.
10. Reyhan Awaliyah Faza yang telah membantu saya dengan berangkat ke Singapura dan membawa komponen tersebut ke Indonesia.
11. Hadi Lizikri Al Azmi, ST. dan Ibunda Mas Hadi Bu Mimi yang telah membantu saya agar komponen dapat diterbangkan langsung ke Surabaya dengan mengikuti penerbangan domestik.
12. Arsya Razak yang telah membantu saya dalam pengerjaan dan memberi saran dalam tugas akhir saya.
13. Yudistira Wahyu Putra, Safarudin, M. Ibnur Kholid yang telah membantu saya selama pengujian alat untuk mencari data.
14. Rekan-rekan bidang studi elektronika yang tidak dapat disebutkan satu-persatu yang telah membantu dan memberikan saran kepada penulis.
15. Rekan-rekan warga, angkatan e55, K-32, Kalpataru dan warung kopi KPK yang telah banyak membantu dan memberi dukungan secara moril terhadap penulis.

Penulis menyadari bahwa masih banyak yang berjasa terhadap tugas akhir penulis yang tidak dapat disebutkan satu persatu sehingga tugas akhir penulis dapat dikembangkan. Oleh karena itu penulis menerima setiap masukan dan kritik yang diberikan. Semoga tugas akhir ini dapat memberikan manfaat bagi banyak pihak.

Surabaya, 18 Januari 2020

Yulizar Edo Pratama Cordova

DAFTAR ISI

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Banyak sekali terjadi kasus kasus survivor atau korban yang berada di dalam hutan dan mencari survivor tersebut secara manual dengan manusia yang menjelajah hutan untuk menemukan lokasi korban. Pengembangan UAV pada saat ini mengembangkan teknologi dimana UAV dapat melakukan penjelajahan untuk mempercepat proses menemukan lokasi korban untuk segera dievakuasi. Pengembangan dilakukan dengan beberapa metode dari secara visual dimana terdapat kamera untuk yang kemudian di stream ke posko atau pusat komando secara live, ada pula yang menggunakan metode pengeras suara yang dibawa oleh UAV bertujuan untuk memberi tanda kepada survivor bahwa ada bantuan datang, dan metode yang dikembangkan akhir akhir ini adalah metode mendeteksi arah sumber suara untuk menentukan lokasi korban.

Banyak penilitian mengenai lokalisasi suara dengan menggunakan *microphone array* dimana cara tersebut dengan memetakan sumber suara dari lebih dari 2 mikrofon yang tertanam pada mikrokontroler yang dibawa oleh UAV. Namun sangat sedikit penelitian yang menggunakan 2 mikrofon untuk memproses lokalisasi suara. Metode menentukan lokasi korban dengan 2 mikrofon sama dengan cara kerja kita menentukan lokasi secara 3 dimensi dan bahkan dapat mengenali keberadaan sumber suara pada posisi belakang dimana tidak terdapat informasi visual[1].

Ketika kedua teknik tersebut digabungkan, maka UAV dapat mendeteksi sinyal suara dari korban yang membutuhkan pertolongan dan berada ditempat yang tidak mampu dijangkau kamera[2] dan pada kondisi malam hari ketika kamera tidak dapat menangkap gambar saat gelap.

1.2 Rumusan Masalah

Rumusan masalah pada tugas akhir ini adalah :

1. Apakah metode ITD dapat digunakan untuk menentukan arah

lokasi sumber suara?
2. Bagaimana cara meredam *noise* yang terbentuk dari *propeller* dan motor sebanyak 4 motor (*quadcopter*)?
3. Bagaimana cara mendeteksi kondisi dimana tidak ada sumber suara ataupun suara keluar dari radius yang dapat ditangkap mikrofon?
4. Bagaimana cara mengirim data secara *wireless* untuk melihat arah sudut sumber suara?

1.3 Batasan Masalah

Batasan Masalah dalam tugas akhir ini adalah :

1. Sumber suara yang dideteksi menggunakan frekuensi tinggi
2. Sumber suara yang dideteksi menggunakan desibel tinggi
3. Jarak sumber suara yang sesuai dengan radius yang dapat ditangkap mikrofon
4. Pengujian dilakukan dengan menjauhkan mikrofon dari motor
5. Mikrofon tanpa menggunakan filter secara analog maupun digital

1.4 Tujuan

Diharapkan dengan adanya pengembangan tugas akhir ini, bisa memberikan :

1. Metode ITD dapat digunakan untuk menentukan lokasi arah sumber suara dengan proses sinyal audio
2. Proses sinyal audio dapat dilakukan saat melakukan penjelajahan pada UAV
3. Data dikirim secara wireless agar UAV dapat terbang jauh dan data dipantau hasilnya

1.5 Metodologi

Dalam pengerjaan tugas akhir ini digunakan metodologi sebagai berikut:

1. Studi Literatur

   Studi literatur berisi serangkaian kegiatan pengumpulan dan pengkajian dasar teori yang relevan dan terpercaya untuk menunjang penulisan tugas akhir ini. Literatur dapat bersumber dari paper, jurnal, artikel, buku, maupun *website* yang bertaraf nasional dan internasional, serta dari hasil konsultasi dengan dosen pembimbing. Namun, terdapat satu sumber rujukan

utama yang telah dilakukan yakni IEEE Explore.

2. Perancangan dan Perakitan

   Merancang gambaran umum *drone* yang akan dikembangkan, mulai dari sisi mekanik, elektronik, perangkat pendeteksi suara, dan algoritma program yang akan digunakan. Bagian mekanik harus memiliki ukuran yang presisi dan beban yang akan disesuaikan dengan keempat motor agar mendapatkan daya angkat yang maksimal serta menimbang seluruh beban perangkat pendeteksi suara. Merancang program agar dapat mempertahankan ketinggian dan melakukan pencarian sumber suara. Data dikirim secara *wireless*.

   Memasukkan program algoritma kedalam perangkat pendeteksi suara dengan menggunakan metode ITD dan menggunakan mikrofon MEMS (*Micro-Electro-Mechanical System*).

3. Tahap Pengujian

   Tahap ini setelah perangkat semua telah terpasang dan program serta metode telah menjadi satu maka *drone* akan diuji coba terlebih dahulu untuk terbang dengan ketinggian tertentu dan menahan ketinggian dengan durasi 10 menit atau lebih. Jika tidak ada masalah teknis baru *drone* akan diukur seberapa besar tingkat noise yang dihasilkan dari keempat motor untuk menentukan penyaringan frekuensi agar tidak terganggu saat perangkat sedang mendeteksi suara. Sumber suara (*beacon*) akan dirubah rubah letaknya agar mendapatkan pembacaan sudut yang berbeda-beda.

4. Analisa dan Evaluasi

   Analisa dilakukan pada saat data yang dikirim secara *wireless* telah masuk dan pembacaan sudut dapat ditampilkan. Lalu data yang masuk akan dicocokkan dengan keberadaan sumber suara secara manual yang diukur dari dimana letak *drone* terbang. Apabila terjadi kesalahan pembacaan maka akan dilakukan tahap evaluasi agar tingkat pembacaan dapat menemui titik akurasi tertinggi dan evaluasi terhadap *drone* akan dilakukan apabila terjadi malfungsi.

5. Penyusunan Laporan

Tahap penyusunan laporan merupakan tahap terakhir dari proses pengerjaan tugas akhir ini. Laporan berisi seluruh hal yang berkaitan dengan tugas akhir yang telah dikerjakan yaitu meliputi pendahuluan, studi literatur, tinjauan pustaka, perancangan dan pembuatan sistem, pengujian dan analisa, serta penutup.

1.6 Sistematika Penulisan

BAB I: PENDAHULUAN

Bab ini meliputi penjelasan latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi.

BAB II: TINJAUAN PUSTAKA DAN TEORI PENUNJANG

Bab ini menjelaskan tentang teori penunjang dan literatur yang dibutuhkan dalam pengerjakan tugas akhir ini. Dasar teori yang menunjang meliputi metode lokalisasi suara, dan cara peredaman dari *noise* motor dan *propeller*. Bagian ini memaparkan mengenai beberapa teori penunjang dan beberapa literatur yang berguna bagi pembuatan Tugas Akhir ini.

BAB III: PERANCANGAN SISTEM

Bab ini menjelasakan secara rinci tentang perencanaan sistem baik *hardware* (*hardware*) maupun *software* (*software*) sistem lokalisasi suara yang dikerjakan.

BAB IV: PENGUJIAN

Pada bab ini akan menjelaskan hasil uji coba sistem beserta analisanya.

BAB V: PENUTUP

Bagian ini merupakan bagian akhir yang berisikan kesimpulan yang diperoleh dari pembuatan Tugas

1.7 Relevansi

Lokalisasi suara menggunakan mikrokontroller STM32F429 berbasis *mini osciloscope* dimana *library* yang tersedia pada mikrokontroller tersebut digunakan untuk mencetak nilai dari 2 sensor mikrofon analog secara bersamaan dan melihat sinyal secara langsung pada *LCD*

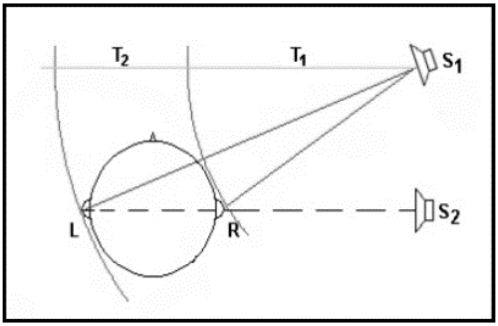
yang ada pada perangkat STM32F429 secara digital. Menghindari sinyal yang termodulasi dan bertumbuk dengan *noise* dari motor dan *propeller* maka teknik yang digunakan adalah Mikrofon MEMS (*Micro-Electro-Mechanical System*) dimana kebal terhadap suara suara bising yang tidak diinginkan atau *noise*.

Halaman Ini Sengaja Dikosongkan

BAB II
DASAR TEORI

2.1 *Sound Localization*

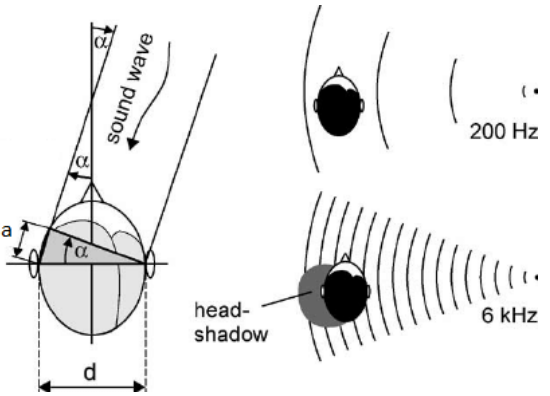
Sound localization adalah proses menentukan lokasi sumber suara berdasarkan beberapa observasi yang diterima pada sinyal suara[3]. Terdapat beberapa metode dalam proses penentuan lokasi pada *sound localization,* salah satunya adalah metode *Time Difference of Arrival* (TDOA). Metode ini terinspirasi dari cara kerja kedua telinga manusia seperti yang diilustrasikan pada Gambar II.1 dalam menerima suara dari posisi tertentu, menggunakan 2 buah atau lebih mikrofon sebagai penerima input suara lalu membedakan waktu datangnya suara dari sumber menuju masing-masing mikrofon. Jika 2 buah mikrofon dipasang pada jarak tertentu, dan saat sumber suara berjarak lebih dekat pada 1 mikrofon daripada mikrofon lainnya, akan terdapat perbedaan waktu sampai antara mikrofon yang lebih dekat dengan yang lebih jauh, perbedaan ini yang selanjutnya digunakan untuk perhitungan sudut arah datangnya sumber suara. Metode ini dapat juga disebut dengan metode *Interaural Time Difference* (ITD)[3].



Gambar II.1 Ilustrasi perbedaan waktu sampai (TDOA) antara sumber suara dengan telinga manusia

Untuk menentukan lokasi sumber suara pada lapangan dibutuhkan perhitungan azimut antara sumber suara dengan

sensor suara. Azimut yang dihasilkan disini mempresentasikan perbandingan lokasi antara sumber suara dengan sensor suara dengan sudut 0^0 sebagai posisi sumber suara tepat di depan sensor suara sesuai dengan yang ditampilkan pada gambar II.1. Sudut ini didapat dari perhitungan TDOA dari sinyal yang diterima oleh keuda sensor suara. Gambar II.2 menunjukkan orientasi sudut posisi depan sumber suara terhadap telinga manusia.



Gambar II.2 Perhitungan Sudut Menggunakan Prinsip Trigonometri

Pencarian azimut pada metode ITD ini pada dasarnya menggunakan prinsip trigonometri dengan nilai d pada gambar II.2 sebagai jarak antara kedua mikrofon yang diibaratkan seperti telinga manusia dan nilai a sebagai jarak yang didapat dari kecepatan suara yang dikalikan dengan waktu jeda pada kedua mikrofon.

Nilai a disini merupakan hasil kali dari kecepatan suara dengan waktu jeda dari kedua mikrofon sesuai dengan persamaan (2.1).

$$length = t \,.\, V_{sound} = (\Delta s \,.\, t_s) \,.\, V_{sound} \qquad (2.1)$$

Nilai t pada persamaan (2.1) adalah waktu jeda yang

digunakan untuk mencari nilai sudut dimana t_s adalah waktu sampling per *array* dan Δs adalah hasil *delay* yang dihitung oleh metode *cross-correlation*, nilai disini adalah kecepatan rambat suara pada udara sebesar 384 m/s pada suhu sebesar 24^0C. Setelah itu didapat nilai a dan d yang hasilnya dapat digunakan untuk menghitung sudut arah sumber suara dengan hukum sinus sesuai pada persamaan (2.3) dan persamaan (2.4).

$$\text{Sin } \theta = \frac{a}{d} \therefore \theta = \text{Sin}^{-1} \frac{a}{d} \quad (2.3)$$

$$\theta = \text{Sin}^{-1} \left(\frac{(\Delta s \,.\, ts) \,.\, Vsound}{d} \right) \quad (2.4)$$

2.1.1 *Cross-correlation*

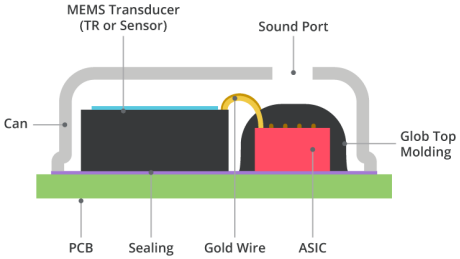
Salah satu variabel yang digunakan untuk mencari nilai Δs pada persamaan (2.1) didapat menggunakan metode *cross-correlation*. Metode ini adalah cara untuk mencari perbedaan dari kedua sinyal, pada kasus ini yang dicari adalah perbedaan fasanya sehingga didapat jeda atau perbedaan waktu sampai sumber suara menuju kedua mikrofon.

$$Corr(k) \equiv \sum_{n=-\infty}^{\infty} (g(n-k) - \mu_g)(h(n) - \mu_h) \quad (2.5)$$

Fungsi *cross-correlation* disini didefinisikan pada persamaan (2.5). Dimana $g(n)$ dan $h(n)$ adalah dua buah sinyal yang akan dicari beda fasanya, μ_g dan μ_h masing-masing adalah *mean* atau rata-rata dari $g(n)$ dan $h(n)$ untuk menormalisasi sinyal agar kebal terhadap *noise*.

2.2 Mikrofon MEMS

Mikrofon MEMS (*Micro-Electro-Mechanical System)* merupakan mikrofon yang ditempatkan pada PCB dan dilindungi dengan penutup mekanis. Lubang kecil dibuat untuk memungkinkan suara masuk ke dalam mikrofon dan ditetapkan sebagai penutup atas. Komponen MEMS dirancang dengan diafragma mekanis dan struktur pemasangan yang dibuat pada dudukan semikonduktor.



Gambar II.3 Mikrofon MEMS

Diafragma MEMS membentuk kapasitor dan gelombang tekanan suara menyebabkan pergerakan pada diafragma. Mikrofon MEMS biasanya berisi semikonduktor yang berfungsi sebagai *preamplifier audio*, yang mengubah kapasitansi MEMS menjadi sinyal listrik. Keluaran dari mikrofon MEMS dapat berupa sinyal analog yang kemudian dapat di konverter ADC (*analog-to-digital*).

Mikrofon MEMS memiliki keunggulan dimana ukurannya yang kecil untuk mengurangi penggunaan area PCB. Keluaran dari MEMS dengan impedansi yang relatif rendah, dengan kelebihan tersebut dapat digunakan pada ruang yang bising elektrik sekalipun. MEMS dapat mengurangi *noise* yang tidak diinginkan yang diakibatkan dari getaran mekanis[4]. Mikrofon MEMS memiliki SNR (*Signal to Noise Ratio*) tinggi, konsumsi daya rendah[5].

2.3 Mikrokontroler STM32F429

STM32F429/439 termasuk dalam keluarga mikrokontroler 32 bit dengan performa dari Cortex-M4 dengan kecepatan proses sampai 180 MHz dengan daya statis yang rendah dibandingkan versi sebelumnya seperti STM32F405/07/15/17. Mikrokontroler STM32F429-Discovery ini cukup menarik, karena kompak dan lengkap dalam satu kemasan papan pengembang (development board) yang sudah termasuk debugger, tombol, LED, gyro sensor, SDRAM dan yang cukup menarik adalah LCD TFT 2,4 inch. Firmware bawaan dari mikrokontroler ini berupa sistem GUI (Graphical User Interface) berbasis STemWin dan memiliki

contoh aplikasi yang bisa ditampilkan di LCD TFT 2.4 inch. Tentu saja kita juga dapat mengganti firmware ini sesuai dengan kebutuhan kita. Pengembangan aplikasi dan pemrograman dapat dilakukan dengan software antara lain: Keil ™ , Em:block(EmBitz) dan bisa juga dengan CooCox IDE[6].

Gambar II.4 *Board STM32F429 Discovery*

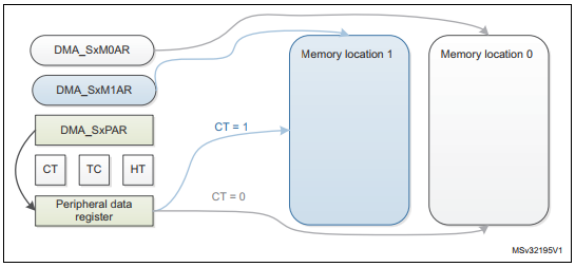
Selain fitur tersebut terdapat juga beberapa kelebihan mikrokontroler STM32F429 dalam operasi pemrosesan sinyal, fitur tersebut adalah *Direct Memory Access* (DMA), DMA adalah sistem yang memiliki arsitektur *bus matrix* dengan beberapa *layer*, dan memori sistem yang berkontribusi dalam memberikan *bandwidth* data yang tinggi serta membuat waktu respon *software* memiliki *latency* atau jeda yang rendah.

Penggunaan DMA memungkinkan untuk proses data secara *background* tanpa intervensi dari prosesor utama Cortex-Mx. Pada operasi ini, prosesor utama dapat menjalankan tugas lain dan hanya dapat di-*interrupt* ketika blok data sudah penuh dan siap untuk diproses.

Data dengan jumlah besar dapat ditransfer tanpa beban besar terhadap perfoma sistem, DMA disini biasanya digunakan untuk implementasi *central data buffer storage* untuk modul periferal berbeda. Solusi ini lebih murah pada bagian silicon dan konsumsi *power* dibanding solusi dimana tiap periferal mengharuskan untuk implementasi *local data storage* sendiri.

Kontroler DMA STM32F429 memanfaatkan kelebihan dari sistem *multi-layer bus* ini untuk dapat memastikan *latency* rendah pada saat transfer DMA dan pada saat proses eksekusi atau deteksi *interrupt* dari prosesor utama.

Salah satu mode transfer pada DMA adalah mode *double buffer* yang memiliki cara kerja mirip seperti transer DMA *single buffer*, dengan perbedaan yaitu pada *pointer* memori yang digunakan sebanyak 2. Ketika mode *double buffer* diaktifkan, *circular mode* secara otomatis aktif dan pada saat akhir dari masing-masing transaksi (DMA_SxNDTR register mencapai 0) *pointer* dari memori akan bergantian. Secara ilustrasi dapat digambarkan prosesnya seperti gambar II.5[1]



Gambar II.5 Mode *Double Buffer*

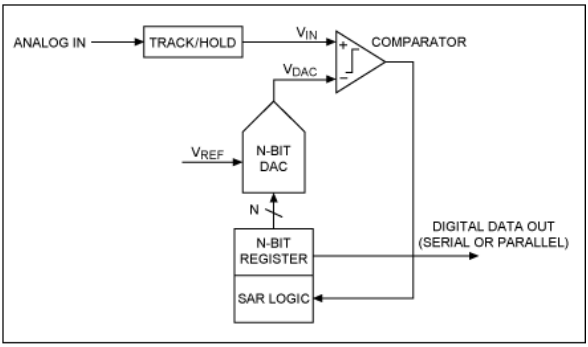
2.3.1 Analog to Digital Converter (ADC) STM32F429

ADC merupkan pengubah besaran analog yang berupa tegangan analog menjadi besaran digital supaya dapat diproses secara digital. Mikrokontroler STM32F4xx memiliki 3 ADC dengan masing-masing ADC memiliki 19 kanal. Sembilan belas kanal ADC ini terbagi menjadi 16 kanal ADC eksternal dan 3 kanal ADC internal.

Tipe ADC yang digunakan pada mikrokontroler ini adalah tipe *Successive-approximation-register* (SAR) ADC, tipe ini banyak digunakan di pasaran dikarenakan kemampuannya dalam memproses ADC dengan *sampling rate* tinggi bernilai hingga 5 Megasamples (Msps) dengan resolusi dari 8 ke 18 bits, selain itu penggunaan daya pada ADC ini juga rendah dengan bentuk yang relatif kecil. 14

Pada intinya tipe ADC ini mengimplementasi algoritma

binary search. Maka dari itu, ketika sirkuit internal mungkin beroperasi pada kecepatan beberapa megahertz (MHz), *sample rate* pada ADC hanya sebagian dari angka tersebut dikarenakan algoritma *successive-approximation*.



Gambar II.6 Arsitektur N-bit SAR ADC

Selanjutnya pada Gambar II.6 ditunjukkan contoh konversi ADC 4 bit. Axis Y (dan garis tebal pada gambar) menunjukkan tegangan output DAC. Pada contoh diatas komparasi pertama menunjukkan bahwa V_{IN} lebih kecil dari V_{DAC}, maka bit 3 diset menjadi 0 dan nilai DAC diset menjadi 0100 lalu komparasi kedua dilakukan. Saat V_{IN} lebih besar dari V_{DAC}, bit 2 tetap bernilai 1 lalu nilai DAC diset kembali menjadi 0110, komparasi ketiga selanjutnya dilakukan, bit 1 diset menjadi 0 dan nilai DAC diset menjadi 0101 untuk komparasi terakhir. Akhirnya nilai bit 0 tetap 1 karena V_{IN} lebih besar dari V_{DAC}.

Dapat dilihat bahwa 4 periode komparasi dibutuhkan untuk ADC 4 bit. Maka dari itu SAR ADC N-bit akan membutuhkan komparasi sebanyak N periode dan tidak akan siap untuk komparasi berikutnya sampai komparasi yang sedang dijalankan selesai. Hal ini menjelaskan kenapa SAR ADC memiliki konsumsi daya yang rendah, tetapi kecepatan yang dilakukan tidak bisa lebih dari *sampling* pada frekuensi lebih dari MHz pada resolusi 14 ke 16 bits[1].

2.4 Pengiriman Data

Untuk mengirim data secara *wireless* diperlukan perangkat yang menerima data dari STM32F429 yang kemudian ditransmit menuju perangkat keras yang lain untuk melihat data sudut dari sumber suara.

2.4.1 NodeMCU V 1.0 ESP8266

Merupakan modul untuk pengembangan dari modul *platform* IoT (*Internet of Things*) keluarga ESP8266 tipe ESP-12. Modul ini menyerupai *platform* Arduino, tetapi yang membedakan adalah dikhususkan untuk "*connected to internet*".

Gambar II.7 *Board NodeMCU V.1.0 ESP8266*

NodeMCU V.1.0 merupakan pengembangan dari versi 0.9 dimana ukuran *board* diperkecil sehingga memperkecil penggunaan area pada PCB. Terdapat PIN yang dikhususkan untuk komunikasi SPI (*Serial Peripherial Interface*) dan PWM (*Pulse With Modulation*) yang tidak ada pada versi sebelumnya.

2.4.2 Bluetooh Module HC-05

Module Bluetooth HC-05 adalah module komunikasi *wireless* yang beroperasi pada frekuensi 2.4 GHz dengan dua pilihan mode konektivitas. Pertama sebagai *slave* atau *receiver* data saja, kedua sebagai master atau sebagai *transceiver*[9].

Gambar II.8 *Module Bluetooth HC-05*

Jangkauan jarak efektif module ini saat terkoneksi dalam range 10 meter, dan jika melebihi dari range tersebut maka kualitas konektivitas akan semakin kurang maksimal[9].

2.5 *Unmanned Aerial Vehicle*

Untuk membawa perangkat keras yang digunakan maka kendaraan yang membawanya adalah UAV atau drone dengan spesifikasi.

2.5.1 Frame F450

Gambar II.9 F450 Frame

Fitur:

- 4 Flame Wheel
- 1 Top Plate
- 1 Bottom Plate
- Material yang bagus dan tahan banting
- Terdapat board didalam bottom plate sebagai distribusi power

2.5.2 Ardupilot Mega 2.8

Gambar II.10 Ardupilot Mega 2.8

Ini adalah APM 2.8 modul autopilot versi ini tidak memiliki kompas pesawat, yang membuat versi ini ideal untuk digunakan dengan multicopters dan penemu. Ini revisi dewan tidak memiliki kompas pesawat, yang dirancang untuk kendaraan (terutama multicopters dan penemu) di mana kompas harus ditempatkan jauh dari kekuasaan dan motor sumber mungkin untuk menghindari gangguan magnetik. (Pada pesawat sayap tetap itu sering lebih mudah untuk me-mount APM cukup jauh dari motor dan ESCs untuk menghindari gangguan magnetik, jadi ini adalah tidak penting, tapi APM 2.8 memberikan lebih banyak fleksibilitas dalam posisi itu dan merupakan pilihan yang baik bagi mereka, juga) . Ini dirancang untuk digunakan dengan 3DR uBlox GPS dengan Kompas (lihat opsi di bawah), sehingga GPS Unit / Kompas dapat dipasang jauh dari sumber-sumber kebisingan dari APM sendiri. APM 2.8 adalah sistem autopilot open source yang lengkap dan teknologi laris yang memenangkan penghargaan bergengsi 2012 Outback Tantangan UAV kompetisi. Hal ini memungkinkan pengguna untuk mengubah apapun tetap, sayap putar atau kendaraan multirotor (bahkan mobil dan kapal) menjadi kendaraan sepenuhnya otonom; mampu melakukan diprogram misi GPS dengan titik arah. Tersedia dengan konektor atas atau samping[7].

2.5.3 Motor Brushless BR2212 Racerstar dan *Propeller* 9inch

Gambar II.11 Motor Brushless BR2212 Racerstar dan *Propeller* 1045

Brand	: Racerstar
Nama	: BR2212 brushless motor
KV	: 1000
Operating Voltage	: 2-4S
Weight	: 52g
Recommended Prop	: 1045 (Diameter 10inch Pitch 4.5 inch)

2.5.4 ESC SimonK Brushless

Gambar II.12 ESC SimonK

ESC (Electronic Speed Control) Brushless 2-4s (Mosfet Toshiba TPCA 8057)

Brand	: SimonK
Maksimum Current	: 30A
BEC Output	: 5V, 3A
Operating Battery	: LiPo 2-4s, Ni-MH 4-12 NiMH
Berat	: 25 g

2.5.5 GPS Neo 7M

Gambar II.13 modul GPS

Brand: uBlox Neo-7M

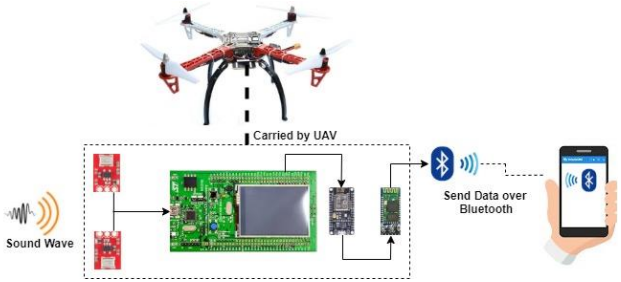
- Presisi yang tinggi GPS dengan kompas
- 38.400 Baudrate
- 10 Hz update rate
- Power dengan LED
- I2C EEPROM storage

BAB III
PERANCANGAN SISTEM

Pada bab ini menjelaskan tentang perancangan dan implementasi sistem meliputi arsitektur sistem, perancangan hardware, software, dan realisasi alat.

3.1 Perancangan Sistem

Sistem terdiri dari 2 buah mikrofon MEMS sebagai input dari sensor suara dan amplifier sebagai penguat sinyal suara serta mikrokontroler sebagai penghitung input yang diterima oleh sensor, lalu mikrokontroler memberi output data serial berupada data sudut pembacaan kepada NodeMCU yang nantinya akan dikirim ke Handphone atau Laptop secara *wireless* dengan media *bluetooth*. Perangkat mikrofon MEMS, STM32F429, NodeMCU, dan *bluetooth* diangkut oleh UAV Berikut adalah ilustrasi diagram sistem yang dibuat:



Gambar III.1 Skema Urutan Sistem Pendeteksi Arah Sumber Suara

Pada Gambar III.1 ditunjukkan bahwa sumber suara merambat melalui udara yang akan ditangkap oleh kedua mikrofon dengan jarak tertentu, lalu akan terdapat perbedaan waktu sampai (TDOA) dari sumber suara menuju kedua mikrofon karena jarak dari sumber suara dengan kedua mikrofon berbeda, jika mikrofon 1 lebih dekat dari sumber suara daripada mikrofon 2, maka waktu sampainya suara dari sumber suara menuju mikrofon 1 akan lebih cepat dibanding mikrofon 2,

begitu juga sebaliknya, setelah perbedaan waktunya didapat, rumus perhitungan sudut arah sumber suara juga dapat dihitung.

Pemrosesan sistem setelah menerima input suara menggunakan mikrokontroler STM32F429, suara input dari mikrofon dibaca oleh ADC pada mikrokontroler sehingga sinyal analog suara diubah menjadi nilai digital yang dapat digunakan untuk proses perhitungan perbedaan waktu dan arah sumber suaranya, setelah sudutnya didapat mikrokontroler akan mengirim data serial ke NodeMCU dan dikirim melalui *bluetooth*.

3.2 Perancangan Hardware

Pada tugas akhir ini hardware yang digunakan adalah 2 mikrofon MEMS, mikrokontroler STM32F429, UAV *Quadcopter*, NodeMCU, *module Bluetooth* HC-05.

Gambar III.2 Perangkat yang telah dirakit menjadi satu

3.2.1 *Unmanned Aerial Vehicle Quadcopter*

UAV dalam penelitian ini menggunakan empat motor *Brushless* BR2212 Racerstar dan empat buah ESC dengan *propeller* 1045, menggunakan modul GPS Neo-7M dengan karakteristik tersebut UAV ini disebut *Quadcopter* dan menggunakan orientasi *compass* secara eksternal. UAV ini bertujuan untuk membawa seluruh perangkat lokalisasi suara dan sebagai sumber tegangan utama untuk perangkat tersebut.

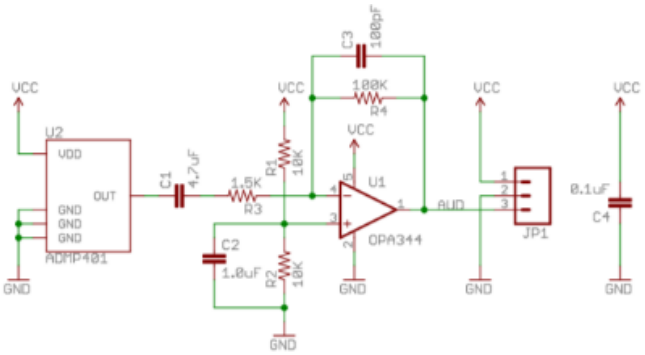
Gambar III.3 UAV membawa perangkat lokalisasi suara

Sumber tegangan UAV tersebut dari baterai lipo yang disambungkan ke ArduPilot Mega 2.8 dan pin (+,-) pada ArduPilot Mega 2.8 terhubung dengan VIN dan GND pada seluruh perangkat yang membutuhkan dan motor dari *quadcopter* terhubung dengan ArduPilot Mega 2.8. mode terbang yang digunakan ada 3 mode yaitu *AtHold* mode mempertahankan ketinggian, *Loiter* mode mempertahankan ketinggian dan posisi dengan orientasi koordinat GPS, dan *Stabillize* mode terbang kontrol penuh.

3.2.2 MEMS Microphone ADMP401

ADMP401 adalah mikrofon dengan kualitas tinggi, daya yang dikonsumsi rendah dan sinyal keluaran analog. ADMP401 menggunakan mikrofon dengan sistem *omnidirectional* dimana dapat menangkan suara secara merata dari semua arah.

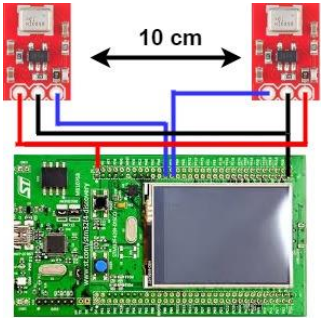
(a)



(b)

Gambar III.4 (a) *Board ADMP401* dan
(b) *Schematic* ADMP401

Output dari mikrofon yaitu pin AUD dimana jika tidak ada suara yang masuk maka perangkat akan mengeluarkan setengah dari Vcc. Amplfier dari mikrofon ini menghasilkan *Peak-to-Peak* sebesar 200mV. Tegangan untuk meyuplai perangkat tersebut memiliki range dari 1,5 sampai 3,3 VDC dengan SNR(*Signal to Noise Ratio*) -62dBA dimana perbandingan komponen sinyal dan komponen *noise* sebesar -62dBA dengan formulasi -SNR=20log $\frac{s}{n}$. Mikrofon ini kebal terhadap getaran mekanis dan *noise* yang dihasilkan motor dan *propeller*.



Gambar III.5 *Wiring* STM32F429 dengan ADMP401

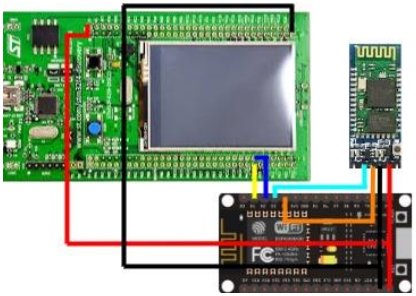
ADMP401 masuk pada pin PA5 dan PA7 dengan *supply voltage* 3,3 VDC dan *ground*, mengacu pada kebutuhan tegangan untuk ADMP401. Mengacu pada penelitian sebelumnya jarak antar mikrofon yang dapat membaca sinyal dengan nilai error dibawah 10 derajat dengan yaitu sebesar 10 cm dengan *error* terkecil pada 7,44% [1] pemasangan mikrofon di sesuaikan dengan penilitian sebelumnya.

3.2.3 Mikrokontroler STM32F429

Channel ADC yang digunakan pada mikrokontroler STM32F429 adalah ADC1 dan ADC2 yang terhubung langsung dengan output pada mikrofon, selain itu *power supply* untuk mikrokontroler dihubungkan dengan VCC dan GND pada ArduPilot Mega 2.8 sebesar 5 VDC.

Gambar III.6 *Wiring Supply STM32F429*

3.2.4 NodeMCU V.1.0 dan Module Bluetooth HC-05



Gambar III.7 *Wiring* NodeMCU dan HC-05

Data yang telah diproses oleh STM32F429 dikirim melalui pin PA10 dan PA9 sebagai (RX, TX) STM32F429 ke pin D2 dan D1 sebagai (RX, TX) NodeMCU V.1.0 ESP8266 dan data dikirim melalui *bluetooth* yang masuk ke pin (RX, TX) HC-05. *Supply voltage* NodeMCU V.1.0 ESP8266 dan

HC-05 dari mikrokontroler sebesar 5 VDC.

Alasan memilih NodeMCU V.1.0 karena kecepatan memproses dari mikrokontroler tersebut 80 MHz dan bekerja pada *baud rate* yang sama dengan STM32F429 yaitu 115200 dan *bluetooth* HC-05 yang bekerja maksimal dengan *baud rate* 9600. Oleh karena itu perlu sebuah jembatan agar data dapat diterjemahkan dan untuk mengirim data tersebut dapat dilakukan pada *baud rate* yang berbeda agar dapat dikirim dan tidak terjadi *flag* (karakter data asli berbeda dengan yang diterima) saat perangkat penerima dapat melihat datanya dengan benar.

3.3 Perancangan software

Mikrokontroler STM32F429 adalah mikrokontroler yang dapat digunakan untuk pengolahan sinyal dengan frekuensi tinggi seperti sinyal suara agar sinyal yang masuk melalui ADC sesuai dengan sinyal yang terbaca pada lapangan, perhitungan pada mikrokontroler ini juga memungkinkan untuk perhitungan untuk prosesi sinyal atau Digital Signal Processing (DSP), oleh karena itu mikrokontroler ini akan mengeluarkan hasil yang lebih akurat dalam prosesi sinyal digital daripada mikrokontroler dengan prosesor lain seperti arduino atau ATMega.

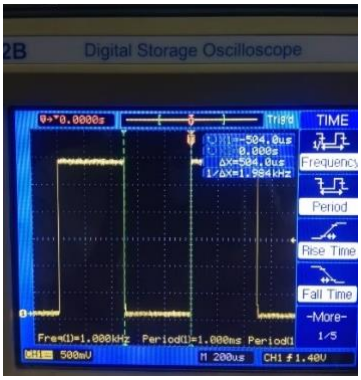
Dalam perancangan software pada proyek ini agar mikrokontroler dapat memproses sinyal suara dengan baik maka diperlukan *benchmark* (melakukan uji langsung melalui media lain) untuk mengetahui *time sampling* yang digunakan untuk mencari variabel pada ITD dan beberapa mode pada sistem ADC dan alokasi memori pada background agar kinerja prosesor jauh lebih efisien.

3.3.1 Time sampling ADC

Untuk Menentukan seberapa besar *time sampling* ADC yang dimiliki oleh STM32F429 perlu dilakukan *benchmark* untuk mengetahui seberapa besar *time sampling* dengan bantuan *oscilloscope*.

Gambar III.8 *Probe* dan *Oscilloscope*

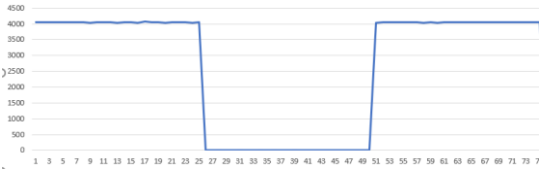
Probe dimasukkan *oscilloscope* untuk mengetahui berapa frekuensi dan berapa waktu yang diperlukan *oscilloscope* untuk membentuk satu gelombang atau satu periode.



Gambar III.9 *Measurement Oscilloscope*

Dari *measurement oscilloscope* bahwa *oscilloscope* menghasilkan sinyal kotak (*square wave*) dengan frekuensi 1KHz dan dalam satu periode diperlukan waktu 1 ms.

Setelah itu bagian output kalibrasi osiloskop dihubungkan ke pin analog di STM32F429 untuk mengukur data *array* dengan melihat berapa banyak data untuk membentuk satu periode.



Gambar III.10 Data *array* dari *Oscilloscope*

Dari data *array* yang telah di plot ke dalam excel menyatakan bahwa panjang *array* untuk membentuk satu gelombang adalah 50 *array* dimana 25 *array* "LOW" dan 25 *array* "HIGH". Dari data tersebut diperoleh *time sampling* dengan cara.

$$T_s = \frac{T_{Oscilloscope}}{N_{array}} = \frac{10\ ms}{50} = 20\ \mu s \qquad (3.1)$$

$$fs = \frac{1}{Ts} = 50\ kHz \qquad (3.2)$$

Dari Persamaan (3.1) dan (3.2) mengacu pada teori Nyquist bahwa frekuensi data harus lebih besar dari frekuensi samplingnya *fsampling* > *fdata* selama frekuensi data masih lebih kecil ataupun dua kali frekuensi data masih belum melewati frekuensi sampling maka data akan tetap valid. Frekuensi sampling dicari karena variabel penting dalam proses ITD.

3.3.2 ADC Dual Mode – Simultaneous

Pada STM32 yang memiliki ADC lebih dari 2 ini terdapat mode dimana kedua ADC dapat bekerja secara bersamaan atau *simultaneously*, proses ini melakukan 2 konversi ADC secara bersamaan sesuai dengan sinkornisasi ADC 1 dan 2. Masing-masing ADC mengonversi susunan channel atau satu channel. Proses konversi dapat dimulai dengan *trigger* eksternal atau melalui *software*. Pada mode ini, hasil konversi ADC1 dan ADC2 akan disimpan pada register data ADC1 (32-bit format). Program ini mendeklarasi pin PA5 sebagai ADC *channel* 1 dan pin PA7 sebagai ADC *channel 2*.

Inisialisasi ADC dan *dual mode*, pada rancangan ini resolusi ADC yang digunakan yaitu 12 bit, dengan mode

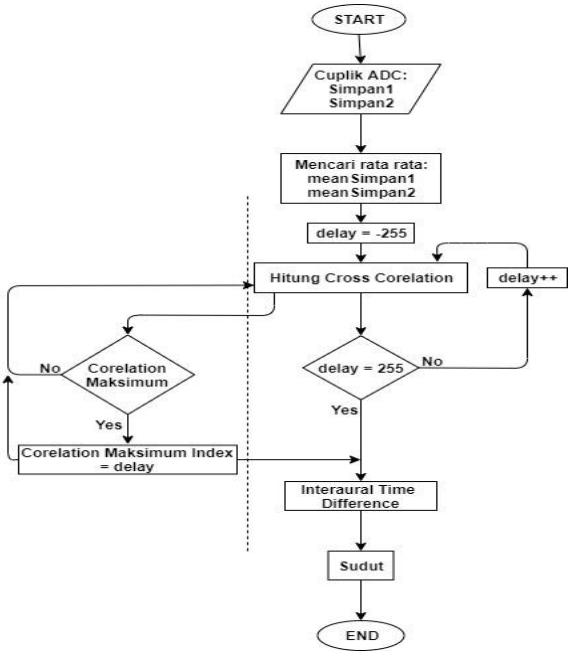
continuous conversion non aktif, *trigger* yang digunakan adalah trigger eksternal yaitu *timer* 2 dengan DMA *continuous request* aktif. Mode yang digunakan adalah ADC *multi mode*, channel yang digunakan yaitu channel 5 dan 7.

3.3.3 Direct Memory Access – Double Buffer

Direct Memory Accessi (DMA) adalah proses yang memungkinkan proses transfer data berjalan pada background atau bekerja tanpa membebani kinerja prosesor. Pada proses ini, prosesor utama dapat melakukan pekerjaan lain dan hanya dapat di-interrupt ketika blok data sudah siap untuk diproses. Data dengan jumlah besar dapat diproses tanpa menyebabkan efek yang besar pada performa sistem.

Mode double buffer disini adalah operasi dimana pengisian data pada memori dilakukan menggunakan 2 buah memory pointer, jadi pengisian data awalnya dilakukan pada memori 1, setelah penuh data akan mengisi memori 2, dengan data pada memori 1 siap untuk diproses. Setelah data memori 2 penuh, data pada memori 2 akan diproses dan pengisian data berpindah kembali pada memori 1. Program ini adalah inisialisasi *buffer* sebagai input pada *cross-corelation* dan sekaligus inisialisasi DMA.

3.3.4 Cross-correlation



Gambar III.11 *Flowchart Interaural Time Difference*

Program cross-correlation pada proyek kali ini digunakan untuk mencari perbedaan waktu sampai kedua sinyal suara pada dua mikrofon dari sumber suara menuju kedua mikrofon, proses ini dilakukan setelah pembacaan mikrofon secara analog yang dikonversi dengan ADC menjadi besaran secara digital dan data masuk ke DMA dan setelah itu dicuplik sepanjang 256 *array* untuk masing masing channel dan diproses pada RAM. Setelah itu data mulai diproses oleh prosesor utama untuk masuk ke perhitungan cross-correlation.

```
// ambil DMA ke RAM
     sprintf (buf, "hasil sampling
ADC[simpan1,simpan2]:");
     p_oszi_send_uart(buf);
     for(n=0;n<256;n++){
       simpan1[n] = ADC_DMA_Buffer_C[n*2];
       simpan2[n] = ADC_DMA_Buffer_C[(n*2)+1];
```

```
        sprintf(buf,"%d,%d",simpan1[n],simpan2[n]);
        p_oszi_send_uart(buf);
        }
      // Cross correlation algorithm
      // 1. hitung mean simpan1 dan simpan2
      uint32_t meanSimpan1,meanSimpan2;
      uint16_t index;
      meanSimpan1 = 0;
      meanSimpan2 = 0;
      for(index=0;index<256;index++){
              meanSimpan1 =
meanSimpan1+simpan1[index];
              meanSimpan2 =
meanSimpan2+simpan2[index];
      }
      meanSimpan1 = meanSimpan1/256;
      meanSimpan2 = meanSimpan2/256;
      // 2.start crosscorrelation
      int delay;
      int j_index;
      int maxDelay;
      int maxValue=0;
      /* Calculate the correlation series */
      for (delay=-255;delay<256;delay++) {
         crossCorrelation[delay+255] = 0;
         for (index=0;index<256;index++) {
           j_index = index + delay;
          if (j_index < 0 || j_index >= 255)
             continue;
          else{
         crossCorrelation[delay+255] +=
(simpan1[index] - meanSimpan1) * (simpan2[j_index]
- meanSimpan2);

if(crossCorrelation[delay+255]>maxValue){
    maxValue = crossCorrelation[delay+255];
                maxDelay = delay;
             }
          }
         }
      }

      // ambil delay
      delta=maxDelay;
      // kirim crossCorrelation
      sprintf (buf, "Hasil Perhitungan cross
correlation:");
      p_oszi_send_uart(buf);
      for(delay=-255;delay<256;delay++){
```

```
            sprintf (buf, "%d",
crossCorrelation[delay+255]);
            p_oszi_send_uart(buf);
     }
     // fromula: derajat = arcsin(
delta*time_sampling*V_sound/jarak_antar_mic)
     derajat =
asinf(((float)delta*20.0/1000000.0)*384/0.11);
     derajat = derajat*180/PI;
     // parsing float agar bisa dikirim serial
     char *tmpSign = (derajat < 0) ? "-" : "";
     float tmpVal = (derajat < 0) ? -derajat :
derajat;
     int tmpInt1 = tmpVal;                  // Get
the integer (678).
     float tmpFrac = tmpVal - tmpInt1;      // Get
fraction (0.0123).
     int tmpInt2 = trunc(tmpFrac * 10000);  //
Turn into integer (123).

     // kirim derajat
     sprintf (buf, "Hasil perhitungan derajat =
%s%d.%02d\n", tmpSign, tmpInt1, tmpInt2);
     p_oszi_send_uart(buf);
//p_oszi_send_uart("END.");
}
```

Gambaran Program *Cross-Corelation* tersebut, program ini memiliki 3 tahap perhitungan dengan rumus mengikuti perhitungan *Cross-Correlation*, hasil dari program ini adalah array data dengan jumlah data 256 untuk masing masing channel. Selanjutnya dicari nilai rata-rata dari array data untuk masing masing channel, nilai rata-rata tersebut untuk dikurangi dengan masing masing channel ADC dan masuk ke tahap selanjutnya menghitung nilai *Cross-Corelation*.

Hasil ADC tiap channel setelah dikurangi dengan rata-rata tiap channel tersebut diolah dengan rumus *Cross-Corelation*, dikarenakan kaidah rumus tersebut -∞ sampai ∞ dan data tidak ada yang dibawah 0 dan lebih dari 255, sehingga diberi syarat jika perkalian untuk dibawah 0 dan lebih dari 255 dilewati. Ketika data dari ADC pertama di nilai [0] dan data dari ADC kedua di nilai [255] perkalian dimulai dan data ADC kedua digeser terhadap ADC pertama sesuai dengan kaidah rumus *Cross-Corelation*.

Selama proses perkalian tersebut hasil dari perkalian

dicek apakah nilai tersebut sudah tertinggi dari nilai sebelumnya dan sesudahnya. Ketika nilai tertinggi ditemukan maka didapatkan maxValue dimana maxValue adalah nilai tertinggi dan didapatkan juga berapa banyak pergeseran yang dicantumkan ke dalam nilai maxDelay dalam bentuk *pointer array*.

Tahap selanjutnya adalah melakukan perkalian dimana *pointer array* tersebut dikalikan dengan *time sampling* ADC sehingga didapatkan nilai jeda perbedaan dari kedua sinyal input. Akhir proses program ini adalah nilai jeda tersebut dikalikan dengan kecepatan suara sebesar 384 m/s dan dibagi dengan jarak mikrofon dan hasilnya di kalikan arcsin atau $\sin^{-1}$ sehingga hasilnya adalah sudut dengan satuan derajat.

3.3.5 Transfer Data

Untuk mengirim data STM32F429 perlu mendeklarasi pin agar dapat mengirim dan diterima oleh pin dari NodeMCU V.1.0 ESP8266 dan selanjutnya dapat dikirim melalui *bluetooth*, untuk itu perlu program untuk mendeklarasikan pin-pin tersebut.

```
typedef enum
{
 COM1 = 0   // COM1 (TX=PA9, RX=PA10)
}UART_NAME_t;
```

Program untuk mendeklarasi pin PA9 sebagai TX dan PA10 sebagai RX untuk STM32F429.

```
void p_oszi_send_uart(char *ptr)
{
    UB_Uart_SendString(COM1,ptr,CRLF);
 }
```

Program untuk memanggil *library* agar data dapat terikirm

```
    sprintf (*character, "%d",*data);
p_oszi_send_uart(*character);
```

Program ini adalah format program untuk mengirim data dimana *character* adalah variabel dengan panjang data yang ditentukan, *data* adalah nilai yang akan dikirim dan "%d" adalah nilai tadi dirubah menjadi nilai *integer*.

```
#include<SoftwareSerial.h>
```

```
#include "HC05.h"

SoftwareSerial ss(4, 5); //RX TX
SoftwareSerial BTSerial(2, 0); //RX TX

void setup() {
  Serial.begin(115200);
  ss.begin(115200);
  BTSerial.begin(9600);
}

void loop() {
  while(ss.available() > 0)
        {
        char data = ss.read();
        BTSerial.write(data);
        Serial.print(data);
  }
}
```

Program ini untuk NodeMCU V.1.0 ESP8266 dimana program tersebut mendeklarasikan pin 4 di GPIO 4 (D2) sebagai RX dan pin 5 di GPIO 5 (D1) sebagai TX dan mengirimkan data secara serial ke HC-05 melalui pin RX dan TX yang telah tersedia pada modul tersebut dan pin 2 di GPIO 2 (D4) sebagai RX dan pin 0 di GPIO 0 (D3) sebagai TX.

BAB IV
PENGUJIAN DAN ANALISIS SISTEM

Program lokalisasi suara dengan metode ITD dengan mencari nilai perkalian tertinggi atau *Cross-Corelation* yang menghasilkan index pergeseran dimana fungsi ini akan menghasilkan nilai sudut.

4.1 Pengujian Hardware

Pada pengujian hardware proyek ini yang diuji adalah uji kestabilan UAV dan mikrofon saat kondisi diterbangkan UAV dan ADC pada mikrokontroler STM32F429, pengujian mikrofon disini mencakup pembacaan saat tidak ada sinyal informasi, saat ada sinyal informasi dengan tanpa gangguan suara *noise* dari motor dan *propeller*, saat *full Throttle* dan saat diterbangkan.

4.1.1 Pengujian kestabilan UAV

Dalam pengujian kali ini adalah menguji kestabilan UAV saat terbang dimana semua perangkat telah digabungkan dan mampu terbang stabil.

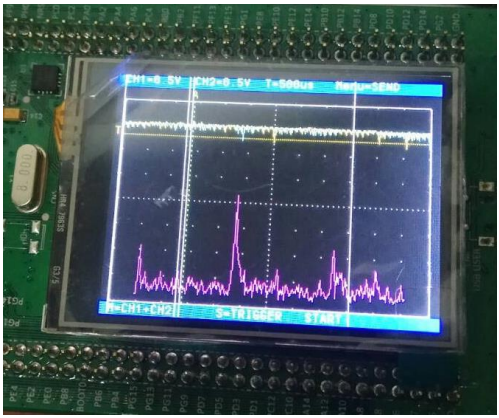
Gambar IV.1 Uji kestabilan terbang

Pada gambar IV.1 tersebut UAV dapat terbang sempurna dan stabil pada mode *AtHold* dimana mode tersebut mempertahankan ketinggian dan pada mode *Loiter* dimana mode tersebut mempertahankan ketinggian sekaligus

mengunci posisi berdasarkan koordinat lokasi GPS yang diterima oleh modul GPS tersebut.

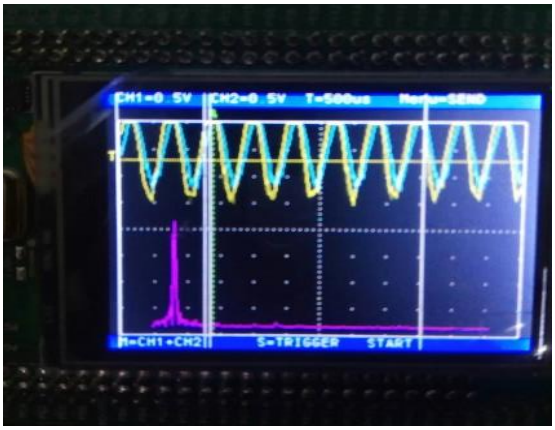
4.1.2 Pengujian Mikrofon pada UAV

Dalam tugas akhir ini hardware yang digunakan adalah mikrofon MEMS dengan module ADMP401 sebagai sensor suara dimana hasil sinyal sudah dikuatkan terlebih dahulu sebelum masuk mikrokontroler. Mikrokontroler STM32F429 sebagai prosesor utama memvisualkan sinyal pada LCD TFT 2.4 inch sebagai penunjuk sinyal secara *real time*.



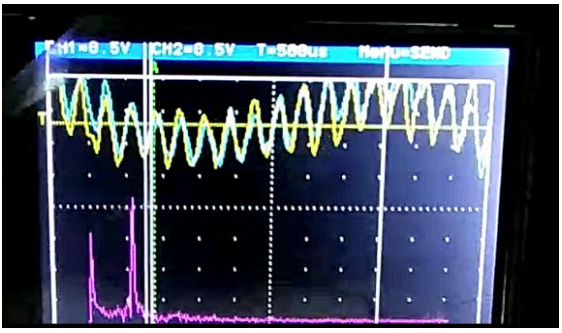
Gambar IV.2 Hasil Pengujian ADMP401 tanpa ada sinyal informasi

Percobaan pertama dilakukan dengan membiarkan sensor suara tidak menerima sinyal apapun. Dapat dilihat dari datasheet menjelaskan bahwa ADMP401 akan memberikan setengah dari tegangan sumber untuk ADMP401.



Gambar IV.3 Hasil Pengujian ADMP401 dengan sinyal informasi tanpa suara noise

Percobaan kedua kali ini dengan memberikan sinyal informasi dengan frekuensi 2KHz. Dapat dilihat hasil dari sinyalnya sangat bersih daripada tidak ada sinyal informasi.



Gambar IV.4 Hasil Pengujian *Full Throttle*

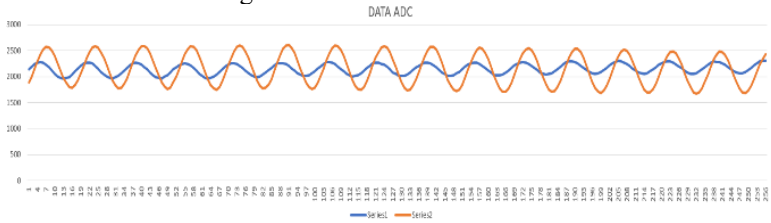
Percobaan ketiga kali ini dengan menaikkan *throttle* pada remote secara penuh yang artinya motor berputar dengan maksimal sehingga UAV terangkat sedikit, dapat dilihat hasilnya dari sinyal termodulasi dan terdapat dua puncak frekuensi dimana sebelah kiri adalah frekuensi dari motor dan yang sebelah kanan adalah sinyal informasi.

4.2 Pengujian Interaural Time Difference

Pengujian kali ini dengan mengolah sinyal suara dari dua sumber yaitu speaker dan peluit yang kemudian menjadi sinyal digital yang telah diterima oleh mikrokontroler yang masuk ke DMA yang kemudian dicuplik kedalam RAM dan diolah dengan metode Cross-Corelation kemudian menggunakan teori trigonometri dan menghasilkan nilai sudut.

4.2.1 Pengujian data ADC STM32F429

Pengujian kali ini mencuplik data yang ditangkap oleh kedua sensor untuk dicetak sepanjang 256 *array* untuk masing masing sensor. Pengujian ADC kali ini menggunakan sudut $+30^0$ dengan frekuensi 3 KHz.

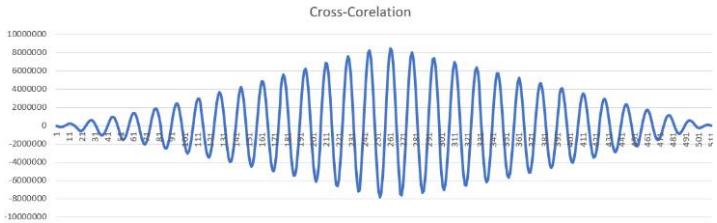


Gambar IV.5 Plotting ADC 256 Array 2 Channel

Dari pengujian tersebut terlihat perbedaan fasa dimana channel 1 *leading* terhadap channel 2 yang nantinya akan dikalikan dengan pengujian terhadap Cross-Corelation.

4.2.2 Pengujian Cross-correleation

Pengujian kali ini adalah menentukan dimana letak atau posisi sebuah sinyal yang berbentuk sama dengan hasil perkalian tertinggi yang dicari melalui pergeseran sinyal penuh untuk secara keseluruhan terhadap sinyal kedua dari sinyal pertama. Data dari STM32F429 ditransfer melalui USBTTL menuju Laptop dan dapat diplotting menjadi data array. Data yang digunakan adalah sama seperti pengujian ADC.

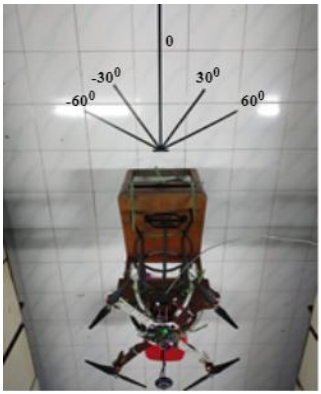


Gambar IV.6 Hasil *Cross-correlation*

hasil dinyatakan bahwa terdapat puncak perkalian tertinggi dimana nanti nilai tersebut berupa besar nilai pergeseran secara index yang akan menjadi variabel penting dalam pengujian ITD.

4.2.3 Pengujian Menyatakan Sudut

Pengujian kali ini dengan menggunakan dua metode dimana sumber pertama speaker dengan frekuensi 3KHz dan sumber kedua peluit, dengan dua mode yaitu UAV tidak menyala dan UAV *full throttle*, dengan 5 sudut dan kondisi terbang.



Gambar IV.6 Metode Pengujian *Indoor*

Gambar IV.7 menunjukkan metode dalam pengujian dibagi menjadi dua sumber suara yaitu peluit dan speaker bluetooth dan pola pengujian menggunakan sudut -60^0, -30^0, 0, 30^0, 60^0. Lalu data akan dikirim melalui bluetooth dan dilihat melalui aplikasi *Bluetooth Terminal*.

Sudut Tujuan (^{0})	Sudut Terukur(^{0})					
-60	-77,81	-77,81	-77,81	-77,81	-77,81	-77,81
-30	-29,25	-29,25	-29,25	-29,25	-29,25	-29,25
0	4,35	4,35	0,35	4,35	0	0
+30	38,92	38,92	38,92	38,92	38,92	38,92
+60	-77,81	-77,81	-77,81	-77,81	-77,81	-77,81

Tabel IV-1 Mode UAV tidak menyala, Suara Speaker

Sudut Tujuan (^{0})	Sudut Terukur(^{0})					
-60	-65,18	-56,91	-20,81	-44,28	-56,91	-14,81
-30	-29,25	-16,21	-33,95	-38,92	-24,76	-24,76
0	0	4,35	-4,35	8,26	-4,35	0
+30	-24,76	-20,43	33,95	29,25	50,17	44,32
+60	65,18	56,91	12,03	44,28	77,81	-16,81

Tabel IV-2 Mode UAV tidak menyala, Suara Peluit

Sudut Tujuan (^{0})	Sudut Terukur(^{0})					
-60	-50.17	-50.17	-50.17	-50.17	-50.17	-50.17
-30	-29,25	-29,25	-44,28	-29,25	-44,28	-44,28
0	0	4,35	0	-4,35	-4,35	0
+30	44.28	29,25	44,28	29,25	44,28	44,28
+60	56,91	56,91	56,91	56,91	56,91	56,91

Tabel IV-3 Mode UAV *full throttle*, Suara Speaker

Sudut Tujuan (^{0})	Sudut Terukur(^{0})					
-60	**-56,91**	**0**	**-4,35**	**-77,81**	**-77,81**	**-56,91**
-30	**0**	**0**	**-33,95**	**-38,92**	**-24,76**	**-24,76**
0	**0**	**4,35**	**-4,35**	**0**	**-4,35**	**0**
+30	**-29,25**	**-14,43**	**12,03**	**29,25**	**0**	**44,32**
+60	**77,81**	**56,91**	**-12,03**	**0**	**77,81**	**-16,81**

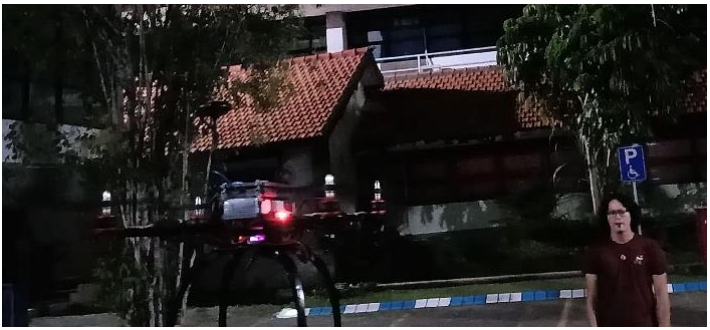
Tabel IV-4 Mode UAV *full throttle*, suara peluit

Pengujian kali ini dengan ruangan terbuka (*outdoor*) dimana pengujian ini tidak menggunakan sudut seperti sebelumnya dikarenakan orientasi GPS yang membuat moncong dari UAV tidak lurus sesuai derajat yang diukur, tetapi pengujian dilakukan dengan membedakan posisi kiri, tengah, dan kanan melalui proyeksi sudut yang dihasilkan. Sumber suara yang digunakan peluit dikarenakan kekerasan suara yang dikeluarkan dan amplitude yang bagus.

Gambar IV.7 Pengujian posisi tengah

Posisi	Sudut Terukur(^{0})					
Tengah	**0**	**-4,35**	**8,26**	**0**	**12,03**	**-4,35**

Tabel IV-5 Mode Terbang posisi tengah



Gambar IV.8 Pengujian posisi kanan

Posisi	Sudut Terukur(^{0})					
Kanan	44,28	12,03	33,95	24,76	44,32	56,91

Tabel IV-6 Mode Terbang posisi kanan

Gambar IV.9 Pengujian posisi kiri

Posisi	Sudut Terukur(^{0})					
Kiri	-29,25	-44,32	-33,95	-8,26	-38,92	-56,91

Tabel IV-7 Mode Terbang posisi kiri

4.3 Analisis

Berikut analisis yang diperoleh dari hasil percobaan dengan beberapa kondisi dan pembacaan sensor:

1. Dalam penelitian ini dilakukan operasi pemrosesan sinyal sehingga membutuhkan kinerja prosesor dengan kecepatan tinggi, maka dari itu digunakan mikrokontroler jenis ARM agar kecepatan pemrosesan data pada prosesor dapat mengatasi input data yang diterima oleh sensor.
2. Tipe mikrofon MEMS yang digunakan tidak terlalu berpengaruh pada hasil sinyal saat diganggu oleh *noise* yang diakibatkan hembusan angin dan motor serta *propeller*.
3. Kondisi terbang secara terbuka akan mengakibatkan banyak suara bertumbuk kearah mikrofon.
4. Tipe sumber suara dari speaker dan peluit memiliki karakteristik sinyal yang berbeda dimana speaker menghasilkan sinyal dengan frekuensi stabil tetapi lemah terhadap amplitude yang terbaca jika mode terbang dan *full throttle* dan tipe sumber suara peluit tidak memiliki frekuensi yang tetap tetapi amplitudonya sangat kuat terbaca saat mode terbang dan *full throttle*.
5. Sudut yang dibaca memiliki nilai yang berbeda dari sudut pembacaan dengan melihat tipe sumber suara dan mode pengujian.
6. Pengujian pada ruangan *indoor* dengan dua mode UAV diperoleh hasil persentase *error* dengan menggunakan sumber suara speaker saat drone tidak menyala adalah 18,76% saat *full throttle* adalah 9 %. Hasil persentase *error* dengan menggunakan sumber suara peluit saat drone tidak menyala adalah 14,71% saat *full throttle* adalah 17,5%.
7. Pada pengujian terbang dengan mode *loiter* dengan 3 posisi arah pembacaan maka hasilnya sesuai dengan nilai sudut yang ditampilkan dengan tanda minus disebelah kiri dan tanda positif disebelah kanan dengan posisi tengah dengan persentase *error* 4,8%

Halaman ini sengaja dikosongkan

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada penelitian ini telah dibuat suatu alat untuk menentukan arah sumber suara berdasarkan metode Interaural Time Differnece menggunakan mikrokontroler STM32F429 pada *unmanned aerial vehicle*. Berdasarkan data hasil uji sistem ini diperoleh beberapa kesimpulan antara lain sebagai berikut:

1. Mikrofon MEMS yang digunakan pada proyek ini dapat membaca sinyal suara peluit speaker secara optimal karena apmlitudonya yang kuat dan dapat membaca sinyal suara speaker secara optimal pada frekuensi 3 KHz keatas atau lebih dari frekuensi drone untuk speaker.
2. Frekuensi rendah yang dihasilkan speaker hanya akan menambah kekerasan suara namun tidak terbaca secara baik oleh sensor dan tertutup frekuensi motor dari drone.
3. Frekuensi tinggi yang dihasilkan speaker memiliki keterbatasan dimana tingkat kekerasan suara yang dihasilkan semakin menurun. Sehingga pengujian data terbang lapangan menggunakan suara peluit karena sinyal dari suara peluit dapat terbaca jelas saat ditiup kencang.
4. Suara peluit memiliki pembacaan yang baik daripada speaker saat mode terbang dan *full throttle*.
5. Penentuan jarak antara mikrofon dengan motor dan *propeller* digunakan untuk mengurangi sinyal yang termodulasi dikarenakan *noise* dari motor, *propeller*, dan angin saat terbang terbuka.
6. Sudut yang dihasilkan memiliki resolusi yang tidak tetap tetapi penanda sebelah kiri dan kanan sudah sangat baik.
7. Pembacaan yang tidak presisi diakibatkan *time sampling* dari STM32F429 yang masih kurang tinggi untuk mencari *delay* dari kedua sinyal.

5.2 Saran

Terkait dengan kendala dan kekurangan dalam penyusunan tugas akhir ini, ada beberapa hal yang dapat penulis sarankan untuk pengembangan selanjutnya. Antara lain sebagai berikut:

1. Penambahan jumlah mikrofon yang digunakan, sehingga sudut yang dapat dideteksi dapat melebihi 180 derajat.
2. Pemakaian tipe mikrofon yang lebih sensitif dan tahan vibrasi sehingga pembacaan suara bisa lebih jauh lagi.
3. Menggunakan motor dan *propeller* yang decibel rendah.
4. Menggunakan motor dengan daya angkat yang lebih tinggi agar drone terbang lebih lama dan tidak boros daya.
5. Agar pembacaan lebih bagus maka menggunakan mikrokontroller dengan *time sampling* yang lebih tinggi agar menambah akurasi pembacaan.
6. Menggunakan metode *clustering* agar suara dapat dibedakan antara suara yang dicari dan suara dari motor dan *propeller* serta *noise* lainnya.
7. Tipe suara yang dihasilkan selanjutnya menggunakan suara murni manusia agar penelitian ini sesuai dengan tujuan penulis.

Demikian saran yang dapat penulis sampaikan. Semoga dapat bermanfaat untuk ke depannya.

DAFTAR PUSTAKA

[1] M. Asfari, M. Rivai, dan T. Tasripan, "*Penentuan Arah Sumber Suara dengan Metode Interaural Time Difference menggunakan Mikrokontroler STM32F4*," J. Tek. ITS, vol. 6, no. 2, Okt 2017.

[2] Satoshi Tadokoro, ""*Listening" drone helps find victims needing rescue in disasters*", Tokyo Institute of Technology, Japan, 2016

[3] Kim, Yong-Eun, Dong-Hyun Su, Jin-Gyung Chung, Xinming Huang, Chul Dong Lee, "Efficient Sound Source Localization Method Using Region Selection", IEEE 2009

[4] Bruce Rose, "*Comparing MEMS and Electret Condenser (ECM) Microphones*",[online],(https://www.cuidevices.com/blog/comparing-mems-and-electret-condenser-microphones, diakses 28 Desember 2019)

[5] EDN, "*Basic Principles of MEMS Microphones*",[online],(https://www.edn.com/basic-principles-of-mems-microphones/, diakses 28 Desember 2019)

[6] Mada Jimmy, "*Mikrokontroler STM32F429 Discovery*",[online],(http://www.madajimmy.com/artikel/blog/68-mikrokontroler-stm32f429-discovery.html, diakses 28 Desember 2019)

[7] Buaya Instrument, "*APM2.8 ArduPilot Mega 2.8"*,[online],(http://buaya-instrument.com/apm26-ardupilot-mega-26-0604410010.html, diakses 28 Desember 2019)

[8] Agus Faudin, "*Apa itu Module NodeMCU ESP8266?*",[online],(https://www.nyebarilmu.com/apa-itu-module-nodemcu-esp8266/, diakses 28 Desember 2019)

[9] Agus Faudin, "*Tutorial Arduino mengakses module Bluetooth HC-05*",[online],(https://www.nyebarilmu.com/tutorial-arduino-module-bluetooth-hc-05/, diakses 28 Desember 2019)

Halaman ini sengaja dikosongkan

LAMPIRAN

Program Sistem Keseluruhan

```
//--------------------------------------------------
// File      : stm32_ub_oszi.c
// Datum     : 24.03.2014
// Version   : 1.6
// Autor     : UB
// EMail     : mc-4u(@)t-online.de
// Web       : www.mikrocontroller-4u.de
// CPU       : STM32F429
// IDE       : CooCox CoIDE 1.7.4
// GCC       : 4.7 2012q4
// Module    : keine
// Funktion  : Oszilloskop
//--------------------------------------------------
unsigned int BufferTotal=0;

//--------------------------------------------------
// Includes
//--------------------------------------------------
#include "oszi.h"
#include "string.h"
#include "stdlib.h"
#include "math.h"

//--------------------------------------------------
//internal function
//--------------------------------------------------
uint32_t p_oszi_hw_init(void);
void p_oszi_sw_init(void);
void p_oszi_clear_all(void);
void p_oszi_draw_background(void);
```

```
void p_oszi_draw_scale(void);
void p_oszi_draw_line_h(uint16_t xp, uint16_t c,
uint16_t m);
void p_oszi_draw_line_v(uint16_t yp, uint16_t c,
uint16_t m);
void p_oszi_sort_adc(void);
void p_oszi_fill_fft(void);
void p_oszi_draw_adc(void);
int16_t oszi_adc2pixel(uint16_t adc, uint32_t
faktor);
void p_oszi_send_data(void);
void p_oszi_send_uart(char *ptr);
void p_oszi_receive_uart(char *ptr);
void p_oszi_send_screen(void);

//Variabel flag
int Flag_MA;
int posisi_tertinggi = 0;
float derajat = 0;
uint16_t simpan1[256];
uint16_t simpan2[256];
int32_t crossCorrelation[511];
int delta;

//max frequency selector variabel
uint16_t FFT_UINT_DATA_MAX;
uint32_t FFT_Index_MAX;


//-----------------------------------------------
----------------
// header untuk transfer BMP
// (ditetapkan sebagai layar lengkap (320x240)
dalam mode landscape)
//-----------------------------------------------
----------------
uint8_t BMP_HEADER[BMP_HEADER_LEN]={
0x42,0x4D,0x36,0x84,0x03,0x00,0x00,0x00,0x00,0x0
0, // ID=BM, Filsize=(240x320x3+54)
```

```
0x36,0x00,0x00,0x00,0x28,0x00,0x00,0x00,
// Offset=54d, Headerlen=40d
0x40,0x01,0x00,0x00,0xF0,0x00,0x00,0x00,0x01,0x0
0, // W=320d, H=240d (landscape)
0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x84,0x03,0x0
0, // 24bpp, unkomprimiert, Data=(240x320x3)
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
// nc
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
// nc



//-----------------------------------------------
----------------
// init dari Oszi
//-----------------------------------------------
----------------
void oszi_init(void)
{
  uint32_t check;

  //-----------------------------------------------
-
  // Hardware init
  //-----------------------------------------------
-
  check=p_oszi_hw_init();
  //p_oszi_send_uart("OSZI 4 STM32F429 [UB]");
  if(check==1) {
    // Touch init error
    UB_LCD_FillLayer(BACKGROUND_COL);
    UB_Font_DrawString(10,10,"Touch
ERR",&Arial_7x10,FONT_COL,BACKGROUND_COL);
    while(1);
  }
  else if(check==2) {
    // Kegagalan ketika Definisi
    UB_LCD_FillLayer(BACKGROUND_COL);
```

```
    UB_Font_DrawString(10,10,"Wrong ADC Array-
LEN",&Arial_7x10,FONT_COL,BACKGROUND_COL);
    while(1);
  }


  //-----------------------------------------
-
  // FFT init
  //-----------------------------------------
-
  fft_init();

  //-----------------------------------------
-
  // Software init
  //-----------------------------------------
-
  p_oszi_sw_init();
  ADC_change_Frq(Menu.timebase.value);
}


//-----------------------------------------
---------------
// start from Oszi (endless loop)
//-----------------------------------------
---------------
void oszi_start(void)
{
  MENU_Status_t status;
//  char buf[10];
  p_oszi_draw_background();
  UB_Graphic2D_Copy2DMA(Menu.akt_transparenz);
  while(1) {
    //-------------------------------------
---
    // tunggu hingga timer GUI kedaluwarsa
    //-------------------------------------
---
```

```
    if(GUI_Timer_ms==0) {
      GUI_Timer_ms=GUI_INTERVALL_MS;
      //-------------------------------------
      // User-Button baca (untuk RUN / STOP)
      //-------------------------------------
      if(UB_Button_OnClick(BTN_USER)==true) {
        status=MENU_NO_CHANGE;
        if(Menu.trigger.mode==2) { // "single"
          if(Menu.trigger.single==4) {
            Menu.trigger.single=5;  // von
"Ready" auf "Stop"
            status=MENU_CHANGE_NORMAL;
          }
        }
        else { // "normal" oder "auto"
          if(Menu.trigger.single==0) {
            Menu.trigger.single=1; // von "Run"
auf "Stop"
            status=MENU_CHANGE_NORMAL;
          }
          else if(Menu.trigger.single==1) {
            Menu.trigger.single=2; // von "Stop"
auf "Weiter"
            status=MENU_CHANGE_NORMAL;
          }
        }
      }
      else {
        //-------------------------------------
        // Tes apakah diperlukan sentuhan
        //-------------------------------------
        status=menu_check_touch();
      }
      if(status!=MENU_NO_CHANGE) {
        p_oszi_draw_background();
        if(status==MENU_CHANGE_FRQ)
ADC_change_Frq(Menu.timebase.value);
        if(status==MENU_CHANGE_MODE) {
          ADC_change_Mode(Menu.trigger.mode);
          if(Menu.trigger.mode!=2) {
```

```
            Menu.trigger.single=0;
          }
          else {
            Menu.trigger.single=3;
          }
          p_oszi_draw_background(); //
menggambar lagi, untuk pembaruan
          ADC_UB.status=ADC_VORLAUF;
          TIM_Cmd(TIM2, ENABLE);
        }
        if(status==MENU_SEND_DATA) {
          p_oszi_draw_background(); //
menggambar lagi, untuk pembaruan
          p_oszi_draw_adc();
          // akan dikirim pada akhirnya
        }
      }

      if(Menu.trigger.mode==1) {
        //-------------------------------------
        // Trigger-Mode = "AUTO"
        // Selalu menggambar ulang layar
        //-------------------------------------
        if(Menu.trigger.single==0) {
          if ((ADC1_DMA_STREAM->CR &
DMA_SxCR_CT) == 0) {
            ADC_UB.trigger_pos=SCALE_X_MITTE;
            ADC_UB.trigger_quarter=2;
          }
          else {
            ADC_UB.trigger_pos=SCALE_X_MITTE;
            ADC_UB.trigger_quarter=4;
          }
          p_oszi_sort_adc();
          p_oszi_fill_fft();
          if(Menu.fft.mode!=0) fft_calc();
          p_oszi_draw_adc();
          ADC_UB.status=ADC_VORLAUF;
          UB_Led_Toggle(LED_RED);
        }
```

```
        else if(Menu.trigger.single==1) {
          // Button "STOP" wurde gedrückt
          // Timer analten
          TIM_Cmd(TIM2, DISABLE);
          if(status!=MENU_NO_CHANGE)
p_oszi_draw_adc();
        }
        else if(Menu.trigger.single==2) {
          // Button "START" wurde gedrückt
          Menu.trigger.single=0;
          ADC_UB.status=ADC_VORLAUF;
          TIM_Cmd(TIM2, ENABLE);
          if(status!=MENU_NO_CHANGE)
p_oszi_draw_adc();
        }
      }
      else if(Menu.trigger.mode==0) {
        //--------------------------------------
        // Trigger-Mode = "NORMAL"
        // Hanya gambar layar setelah peristiwa
pemicu
        //--------------------------------------
        if(Menu.trigger.single==0) {
          if(ADC_UB.status==ADC_READY) {
            UB_Led_Toggle(LED_RED);
            p_oszi_sort_adc();
            p_oszi_fill_fft();
            if(Menu.fft.mode!=0) fft_calc();
            p_oszi_draw_adc();
            ADC_UB.status=ADC_VORLAUF;
            TIM_Cmd(TIM2, ENABLE);
          }
          else {
            // oder wenn Menu verändert wurde

if(status!=MENU_NO_CHANGE)p_oszi_draw_adc();
          }
        }
        else if(Menu.trigger.single==1) {
          // Button "STOP" wurde gedrückt
```

```
          // Timer analten
          TIM_Cmd(TIM2, DISABLE);
          if(status!=MENU_NO_CHANGE)
p_oszi_draw_adc();
        }
        else if(Menu.trigger.single==2) {
          // Button "START" wurde gedrückt
          Menu.trigger.single=0;
          ADC_UB.status=ADC_VORLAUF;
          TIM_Cmd(TIM2, ENABLE);
          if(status!=MENU_NO_CHANGE)
p_oszi_draw_adc();
        }
      }
      else {
        //--------------------------------------
        // Trigger-Mode = "SINGLE"
        // Gambar layar hanya sekali, setelah
peristiwa pemicu
        //--------------------------------------
        if(Menu.trigger.single==3) {
          // Menunggu Trigger-Event
          if(ADC_UB.status==ADC_READY) {
            Menu.trigger.single=4;
            UB_Led_Toggle(LED_RED);
            p_oszi_sort_adc();
            p_oszi_fill_fft();
            if(Menu.fft.mode!=0) fft_calc();
            p_oszi_draw_adc();
          }
          else {
            // atau ketika Menu berubah
            if(status!=MENU_NO_CHANGE)
p_oszi_draw_adc();
          }
        }
        else if(Menu.trigger.single==5) {
          // Button "Reset" ketika ditekan
          Menu.trigger.single=3;
          p_oszi_draw_adc();
```

```
          ADC_UB.status=ADC_VORLAUF;
          TIM_Cmd(TIM2, ENABLE);
        }
        else {
          // atau ketika Menu berubah
          if(status!=MENU_NO_CHANGE)
p_oszi_draw_adc();
        }
      }

      if(GUI.gui_xpos==GUI_XPOS_OFF) {
        // Tanpa GUI => Tanpa Tampilan
Transparan
        UB_Graphic2D_Copy1DMA();
      }
      else {
        // Dengan GUI => Dengan Tampilan
Transparan

UB_Graphic2D_Copy2DMA(Menu.akt_transparenz);
      }

      // Refreh LCD
      UB_LCD_Refresh();

      // event. Pengiriman Data
//      if(Menu.send.data==0) {
        p_oszi_send_data();
//       Menu.send.data!=0;
//      }
    }
  }
}


//-----------------------------------------------
----------------
// init Hardware
//-----------------------------------------------
----------------
```

```
uint32_t p_oszi_hw_init(void)
{
  uint32_t ret_wert=0;

  // Init dari Touch
  if(UB_Touch_Init()!=SUCCESS) {
    ret_wert=1; // Touch error
  }

  // Check Definition
  if(ADC_ARRAY_LEN!=SCALE_W) {
    ret_wert=2; // define error
  }

  // Init Systick
  UB_Systick_Init();

  // InitLEDs
  UB_Led_Init();

  // Init  Button
  UB_Button_Init();

  // Init UART
  UB_Uart_Init();

  // init vom LCD (und SD-RAM)
  UB_LCD_Init();
  UB_LCD_LayerInit_Fullscreen();
  UB_LCD_SetMode(LANDSCAPE);

  // clear all buffers
  p_oszi_clear_all();

  // init from ADC
  ADC_Init_ALL();

  return(ret_wert);
}
```

```
//-----------------------------------------------
----------------
// init Software
//-----------------------------------------------
----------------
void p_oszi_sw_init(void)
{
  //--------------------------------------
  // Pengaturan default
  //--------------------------------------
  Menu.akt_transparenz=100;
  Menu.akt_setting=SETTING_SEND;

  Menu.ch1.faktor=3;       // 1v/div
  Menu.ch1.visible=0;      // 0 = visible, 1 =
true
  Menu.ch1.position=0;

  Menu.ch2.faktor=3;       // 1v/div
  Menu.ch2.visible=0;      // visible=true
  Menu.ch2.position=0;

  Menu.timebase.value=12;  // 100us/div

  Menu.trigger.source=1;   // trigger=CH2
  Menu.trigger.edge=0;     // High-trigger
  Menu.trigger.mode=0;     // auto
  Menu.trigger.single=0;
  Menu.trigger.value_ch1=1024;
  Menu.trigger.value_ch2=1800;

  Menu.cursor.mode=4;      // cursor Off
  Menu.cursor.p1=2048;
  Menu.cursor.p2=3072;
  Menu.cursor.t1=1600;
  Menu.cursor.t2=2000;
  Menu.cursor.f1=750;

  Menu.send.mode=4; // mode CH1+CH2
```

```
  Menu.send.screen=SETTING_TRIGGER;
  Menu.send.data=0;

  Menu.fft.mode=2; // FFT=CH2

  GUI.gui_xpos=GUI_XPOS_OFF;  // GUI
disembunyikan
  GUI.akt_menu=MM_NONE;
  GUI.old_menu=MM_CH1;
  GUI.akt_button=GUI_BTN_NONE;
  GUI.old_button=GUI_BTN_NONE;
}


//-----------------------------------------------
----------------
// membersihkan semua memori
//-----------------------------------------------
----------------
void p_oszi_clear_all(void)
{
  UB_LCD_SetLayer_2();
  UB_LCD_SetTransparency(255);
  UB_LCD_FillLayer(BACKGROUND_COL);
  UB_LCD_Copy_Layer2_to_Layer1();
  UB_LCD_SetLayer_Menu();
  UB_LCD_FillLayer(BACKGROUND_COL);
  UB_LCD_SetLayer_ADC();
  UB_LCD_FillLayer(BACKGROUND_COL);
  UB_LCD_SetLayer_Back();
}


//-----------------------------------------------
----------------
// gambar latar belakang dari Oszi
// (skala, kursor, menu, dll.)
// alamat tujuan di SD-RAM = [MENU]
//-----------------------------------------------
----------------
```

```
void p_oszi_draw_background(void)
{
  UB_LCD_SetLayer_Menu();
  UB_LCD_FillLayer(BACKGROUND_COL);

  // Draw GUI first
  menu_draw_all();
  // then draw the scale and cursor
  p_oszi_draw_scale();

  UB_LCD_SetLayer_Back();
}


//-------------------------------------------------
----------------
// Menampilkan Skala dan Cursor dari Oszi
//-------------------------------------------------
----------------
void p_oszi_draw_scale(void)
{
  uint32_t n,m;
  uint16_t xs,ys;
  int16_t signed_int;

  xs=SCALE_START_X;
  ys=SCALE_START_Y;

  //-----------------------------------------------
-
  // Menggambarkan dari masing-masing titik
  //-----------------------------------------------
-
  for(m=0;m<=SCALE_H;m+=SCALE_SPACE) {
    for(n=0;n<=SCALE_W;n+=SCALE_SPACE) {

UB_Graphic2D_DrawPixelNormal(m+xs,n+ys,SCALE_COL
);
    }
  }
```

```
  //-------------------------------------------
-
  // X-Axis
  //-------------------------------------------
-
  signed_int=SCALE_Y_MITTE+xs;
  p_oszi_draw_line_h(signed_int,SCALE_COL,0);

  //-------------------------------------------
-
  // Y-Axis
  //-------------------------------------------
-
  signed_int=SCALE_X_MITTE+ys;
  p_oszi_draw_line_v(signed_int,SCALE_COL,0);

  //-------------------------------------------
-
  // Umrandung
  //-------------------------------------------
-
  // unterste linie
  UB_Graphic2D_DrawStraightDMA(xs-1,ys-
1,SCALE_W+2,LCD_DIR_HORIZONTAL,SCALE_COL);
  // oberste linie
  UB_Graphic2D_DrawStraightDMA(xs+SCALE_H+1,ys-
1,SCALE_W+2,LCD_DIR_HORIZONTAL,SCALE_COL);
  // linke linie
  UB_Graphic2D_DrawStraightDMA(xs-1,ys-
1,SCALE_H+2,LCD_DIR_VERTICAL,SCALE_COL);
  // rechte linie
  UB_Graphic2D_DrawStraightDMA(xs-
1,ys+SCALE_W+1,SCALE_H+2,LCD_DIR_VERTICAL,SCALE_
COL);


  //-------------------------------------------
-
  // Trigger-Linie (immer Sichtbar)
```

```
  //-----------------------------------------
-
  if(Menu.trigger.source==0) {

signed_int=oszi_adc2pixel(Menu.trigger.value_ch1
, Menu.ch1.faktor);

signed_int+=SCALE_Y_MITTE+SCALE_START_X+Menu.ch1
.position;
    if(signed_int<SCALE_START_X)
signed_int=SCALE_START_X;
    if(signed_int>SCALE_MX_PIXEL)
signed_int=SCALE_MX_PIXEL;


p_oszi_draw_line_h(signed_int,ADC_CH1_COL,1);
    UB_Font_DrawString(signed_int-
3,0,"T",&Arial_7x10,ADC_CH1_COL,BACKGROUND_COL);
  }
  else if(Menu.trigger.source==1) {

signed_int=oszi_adc2pixel(Menu.trigger.value_ch2
, Menu.ch2.faktor);

signed_int+=SCALE_Y_MITTE+SCALE_START_X+Menu.ch2
.position;
    if(signed_int<SCALE_START_X)
signed_int=SCALE_START_X;
    if(signed_int>SCALE_MX_PIXEL)
signed_int=SCALE_MX_PIXEL;


p_oszi_draw_line_h(signed_int,ADC_CH2_COL,1);
    UB_Font_DrawString(signed_int-
3,0,"T",&Arial_7x10,ADC_CH2_COL,BACKGROUND_COL);
  }

  //-----------------------------------------
-
  // Cursor-Linien (nur falls aktiviert)
```

```
  //-----------------------------------------
-
  if(Menu.cursor.mode==1) {
    //-------------------------------
    // Cursor (CH1)
    //-------------------------------
    signed_int=oszi_adc2pixel(Menu.cursor.p1,
Menu.ch1.faktor);

signed_int+=SCALE_Y_MITTE+SCALE_START_X+Menu.ch1
.position;
    if(signed_int<SCALE_START_X)
signed_int=SCALE_START_X;
    if(signed_int>SCALE_MX_PIXEL)
signed_int=SCALE_MX_PIXEL;

    p_oszi_draw_line_h(signed_int,CURSOR_COL,2);
    UB_Font_DrawString(signed_int-
3,312,"A",&Arial_7x10,CURSOR_COL,BACKGROUND_COL)
;

    signed_int=oszi_adc2pixel(Menu.cursor.p2,
Menu.ch1.faktor);

signed_int+=SCALE_Y_MITTE+SCALE_START_X+Menu.ch1
.position;
    if(signed_int<SCALE_START_X)
signed_int=SCALE_START_X;
    if(signed_int>SCALE_MX_PIXEL)
signed_int=SCALE_MX_PIXEL;

    p_oszi_draw_line_h(signed_int,CURSOR_COL,2);
    UB_Font_DrawString(signed_int-
3,312,"B",&Arial_7x10,CURSOR_COL,BACKGROUND_COL)
;
  }
  else if(Menu.cursor.mode==2) {
    //-------------------------------
    // Cursor (CH2)
    //-------------------------------
```

```
    signed_int=oszi_adc2pixel(Menu.cursor.p1,
Menu.ch2.faktor);

signed_int+=SCALE_Y_MITTE+SCALE_START_X+Menu.ch2
.position;
    if(signed_int<SCALE_START_X)
signed_int=SCALE_START_X;
    if(signed_int>SCALE_MX_PIXEL)
signed_int=SCALE_MX_PIXEL;

    p_oszi_draw_line_h(signed_int,CURSOR_COL,2);
    UB_Font_DrawString(signed_int-
3,312,"A",&Arial_7x10,CURSOR_COL,BACKGROUND_COL)
;

    signed_int=oszi_adc2pixel(Menu.cursor.p2,
Menu.ch2.faktor);

signed_int+=SCALE_Y_MITTE+SCALE_START_X+Menu.ch2
.position;
    if(signed_int<SCALE_START_X)
signed_int=SCALE_START_X;
    if(signed_int>SCALE_MX_PIXEL)
signed_int=SCALE_MX_PIXEL;

    p_oszi_draw_line_h(signed_int,CURSOR_COL,2);
    UB_Font_DrawString(signed_int-
3,312,"B",&Arial_7x10,CURSOR_COL,BACKGROUND_COL)
;
  }
  else if(Menu.cursor.mode==3) {
    //-------------------------------
    // Cursor (TIME)
    //-------------------------------
    signed_int=Menu.cursor.t1*FAKTOR_T;
    signed_int+=SCALE_START_Y;
    if(signed_int<SCALE_START_Y)
signed_int=SCALE_START_Y;
    if(signed_int>SCALE_MY_PIXEL)
signed_int=SCALE_MY_PIXEL;
```

```
    p_oszi_draw_line_v(signed_int,CURSOR_COL,2);
    UB_Font_DrawString(215,signed_int-
3,"A",&Arial_7x10,CURSOR_COL,BACKGROUND_COL);

    signed_int=Menu.cursor.t2*FAKTOR_T;
    signed_int+=SCALE_START_Y;
    if(signed_int<SCALE_START_Y)
signed_int=SCALE_START_Y;
    if(signed_int>SCALE_MY_PIXEL)
signed_int=SCALE_MY_PIXEL;

    p_oszi_draw_line_v(signed_int,CURSOR_COL,2);
    UB_Font_DrawString(215,signed_int-
3,"B",&Arial_7x10,CURSOR_COL,BACKGROUND_COL);
  }
  else if(Menu.cursor.mode==4) {
    //------------------------------
    // Cursor (FFT)
    //------------------------------
    signed_int=Menu.cursor.f1*FAKTOR_F;
    signed_int+=FFT_START_Y+1;
    if(signed_int<FFT_START_Y)
signed_int=FFT_START_Y;

if(signed_int>(FFT_START_Y+FFT_VISIBLE_LENGTH))
signed_int=(FFT_START_Y+FFT_VISIBLE_LENGTH);

    p_oszi_draw_line_v(signed_int,CURSOR_COL,2);
    UB_Font_DrawString(215,signed_int-
3,"A",&Arial_7x10,CURSOR_COL,BACKGROUND_COL);
  }
}


//-----------------------------------------------
---------------
// zeichnet eine horizontale Line auf das Oszi-
Gitter
// an "xp", mit Farbe "c" und Mode "m"
```

```
//-----------------------------------------------
----------------
void p_oszi_draw_line_h(uint16_t xp, uint16_t c,
uint16_t m)
{
  uint32_t n,t;

  if(m==0) {
    // Linie : "X----X----X----X----X----X"
    for(n=0;n<=SCALE_W;n+=5) {

UB_Graphic2D_DrawPixelNormal(xp,n+SCALE_START_Y,
c);
    }
  }
  else if(m==1) {
    // Linie : "X-X-X-X-X-X-X-X-X-X"
    for(n=0;n<=SCALE_W;n+=2) {

UB_Graphic2D_DrawPixelNormal(xp,n+SCALE_START_Y,
c);
    }
  }
  else if(m==2) {
    // Linie : "XX---XX---XX---XX---XX"
    t=0;
    for(n=0;n<=SCALE_W;n++) {
      if(t<2)
UB_Graphic2D_DrawPixelNormal(xp,n+SCALE_START_Y,
c);
      t++;
      if(t>4) t=0;
    }
  }
}


//-----------------------------------------------
----------------
```

```
// zeichnet eine vertikale Line auf das Oszi-
Gitter
// an "yp", mit Farbe "c" und Mode "m"
//-----------------------------------------
----------------
void p_oszi_draw_line_v(uint16_t yp, uint16_t c,
uint16_t m)
{
  uint32_t n,t;

  if(m==0) {
    // Linie : "X----X----X----X----X----X"
    for(n=0;n<=SCALE_H;n+=5) {

UB_Graphic2D_DrawPixelNormal(n+SCALE_START_X,yp,
c);
    }
  }
  else if(m==1) {
    // Linie : "X-X-X-X-X-X-X-X-X-X"
    for(n=0;n<=SCALE_H;n+=2) {

UB_Graphic2D_DrawPixelNormal(n+SCALE_START_X,yp,
c);
    }
  }
  else if(m==2) {
    // Linie : "XX---XX---XX---XX---XX"
    t=0;
    for(n=0;n<=SCALE_H;n++) {
      if(t<2)
UB_Graphic2D_DrawPixelNormal(n+SCALE_START_X,yp,
c);
      t++;
      if(t>4) t=0;
    }
  }
}
```

```
//-----------------------------------------------
----------------
// mengurutkan data saluran ADC dari Buffer_A
dan Buffer_B
// ke Buffer_C
// data diurutkan dengan cara yang sama dengan
peristiwa pemicu di tengah
// dari area data (di bagian tengah layar nanti)
//
// Titik pemicu bisa di salah satu dari 4
kuadran
// berasal dari buffer data
// kuadran-1 = paruh pertama buffer-A
// kuadran-2 = paruh kedua buffer-A
// kuadran-3 = paruh pertama buffer-B
// kuadran-4 = paruh kedua buffer-B
//-----------------------------------------------
----------------
void p_oszi_sort_adc(void)
{
  uint32_t n=0;
  uint32_t start=0,anz1=0,anz2=0;
  uint16_t wert;

  if(ADC_UB.trigger_quarter==1) {
    //------------------------------
    // Trigger-Punkt liegt in Q1
    //------------------------------
    anz1=(SCALE_X_MITTE-ADC_UB.trigger_pos);
    start=SCALE_W-anz1;

    //------------------------------
    // Salin Bagian Kiri (?)
    //------------------------------
    for(n=0;n<anz1;n++) {
      wert=ADC_DMA_Buffer_B[(start+n)*2];
      ADC_DMA_Buffer_C[n*2]=wert;
      wert=ADC_DMA_Buffer_B[((start+n)*2)+1];
      ADC_DMA_Buffer_C[(n*2)+1]=wert;
    }
```

```
    //------------------------------
    // Salin bagian kanan (?)
    //------------------------------
    anz2=SCALE_W-anz1;
    start=0;
    for(n=0;n<anz2;n++) {
      wert=ADC_DMA_Buffer_A[(start+n)*2];
      ADC_DMA_Buffer_C[(n+anz1)*2]=wert;
      wert=ADC_DMA_Buffer_A[((start+n)*2)+1];
      ADC_DMA_Buffer_C[((n+anz1)*2)+1]=wert;
    }
  }
  else if(ADC_UB.trigger_quarter==2) {
    //------------------------------
    // Trigger-Punkt liegt in Q2
    //------------------------------
    anz1=SCALE_W-((ADC_UB.trigger_pos-
SCALE_X_MITTE));
    start=SCALE_W-anz1;

    //------------------------------
    // linker Teil kopieren
    //------------------------------
    for(n=0;n<anz1;n++) {
      wert=ADC_DMA_Buffer_A[(start+n)*2];
      ADC_DMA_Buffer_C[n*2]=wert;
      wert=ADC_DMA_Buffer_A[((start+n)*2)+1];
      ADC_DMA_Buffer_C[(n*2)+1]=wert;
    }
    //------------------------------
    // rechter Teil kopieren
    //------------------------------
    anz2=SCALE_W-anz1;
    start=0;
    for(n=0;n<anz2;n++) {
      wert=ADC_DMA_Buffer_B[(start+n)*2];
      ADC_DMA_Buffer_C[(n+anz1)*2]=wert;
      wert=ADC_DMA_Buffer_B[((start+n)*2)+1];
      ADC_DMA_Buffer_C[((n+anz1)*2)+1]=wert;
    }
```

```
  }
  else if(ADC_UB.trigger_quarter==3) {
    //------------------------------
    // Trigger-Punkt liegt in Q3
    //------------------------------
    anz1=(SCALE_X_MITTE-ADC_UB.trigger_pos);
    start=SCALE_W-anz1;

    //------------------------------
    // linker Teil kopieren
    //------------------------------
    for(n=0;n<anz1;n++) {
      wert=ADC_DMA_Buffer_A[(start+n)*2];
      ADC_DMA_Buffer_C[n*2]=wert;
      wert=ADC_DMA_Buffer_A[((start+n)*2)+1];
      ADC_DMA_Buffer_C[(n*2)+1]=wert;
    }
    //------------------------------
    // rechter Teil kopieren
    //------------------------------
    anz2=SCALE_W-anz1;
    start=0;
    for(n=0;n<anz2;n++) {
      wert=ADC_DMA_Buffer_B[(start+n)*2];
      ADC_DMA_Buffer_C[(n+anz1)*2]=wert;
      wert=ADC_DMA_Buffer_B[((start+n)*2)+1];
      ADC_DMA_Buffer_C[((n+anz1)*2)+1]=wert;
    }
  }
  else if(ADC_UB.trigger_quarter==4) {
    //------------------------------
    // Trigger-Punkt liegt in Q4
    //------------------------------
    anz1=SCALE_W-((ADC_UB.trigger_pos-
SCALE_X_MITTE));
    start=SCALE_W-anz1;

    //------------------------------
    // linker Teil kopieren
    //------------------------------
```

```
    for(n=0;n<anz1;n++) {
      wert=ADC_DMA_Buffer_B[(start+n)*2];
      ADC_DMA_Buffer_C[n*2]=wert;
      wert=ADC_DMA_Buffer_B[((start+n)*2)+1];
      ADC_DMA_Buffer_C[(n*2)+1]=wert;
    }
    //------------------------------
    // rechter Teil kopieren
    //------------------------------
    anz2=SCALE_W-anz1;
    start=0;
    for(n=0;n<anz2;n++) {
      wert=ADC_DMA_Buffer_A[(start+n)*2];
      ADC_DMA_Buffer_C[(n+anz1)*2]=wert;
      wert=ADC_DMA_Buffer_A[((start+n)*2)+1];
      ADC_DMA_Buffer_C[((n+anz1)*2)+1]=wert;
    }
  }
}


//-----------------------------------------------
----------------
// mengisi buffer input FFT
// dengan data sampel dari CH1 atau CH2
// (isi sisanya dengan 0)
//-----------------------------------------------
----------------
void p_oszi_fill_fft(void)
{
  uint32_t n,m;

  if(Menu.fft.mode==1) {
    m=0;
    for(n=0;n<FFT_LENGTH;n++) {
      if(m<SCALE_W) {

FFT_DATA_IN[n]=(float32_t)((ADC_DMA_Buffer_C[(m*
2)]-2048.0)/1000.0);
      }
```

```
      else {
        FFT_DATA_IN[n]=0.0;
      }
      m++;
    }
  }
  else if(Menu.fft.mode==2) {
    m=0;
    for(n=0;n<FFT_LENGTH;n++) {
      if(m<SCALE_W) {

FFT_DATA_IN[n]=(float32_t)((ADC_DMA_Buffer_C[(m*
2)+1]-2048.0)/1000.0);
      }
      else {
        FFT_DATA_IN[n]=0.0;
      }
      m++;
    }
  }
}


//-----------------------------------------------
----------------
// merekam data dari dua saluran ADC (dan FFT)
//-----------------------------------------------
----------------
void p_oszi_draw_adc(void)
{
  uint32_t n=0;
  int16_t ch1_wert1,ch1_wert2;
  int16_t ch2_wert1,ch2_wert2;
  int16_t fft_wert1,fft_wert2;

  p_oszi_draw_background();
  UB_LCD_SetLayer_Menu();

  // Nilai awal
```

```
  ch1_wert1=oszi_adc2pixel(ADC_DMA_Buffer_C[0],
Menu.ch1.faktor);

ch1_wert1+=SCALE_Y_MITTE+SCALE_START_X+Menu.ch1.
position;   //Nilainya ditampilin di pixel mana
setelah ditambah scale
  if(ch1_wert1<SCALE_START_X)
ch1_wert1=SCALE_START_X;
  if(ch1_wert1>SCALE_MX_PIXEL)
ch1_wert1=SCALE_MX_PIXEL;

  ch2_wert1=oszi_adc2pixel(ADC_DMA_Buffer_C[1],
Menu.ch2.faktor);

ch2_wert1+=SCALE_Y_MITTE+SCALE_START_X+Menu.ch2.
position;
  if(ch2_wert1<SCALE_START_X)
ch2_wert1=SCALE_START_X;
  if(ch2_wert1>SCALE_MX_PIXEL)
ch2_wert1=SCALE_MX_PIXEL;

  fft_wert1=FFT_UINT_DATA[0];
  fft_wert1+=FFT_START_X;
  if(fft_wert1<SCALE_START_X)
fft_wert1=SCALE_START_X;
  if(fft_wert1>SCALE_MX_PIXEL)
fft_wert1=SCALE_MX_PIXEL;

  //Select Max Frequency
  FFT_UINT_DATA_MAX=0;
  FFT_Index_MAX=0;
    for(n=2;n<FFT_VISIBLE_LENGTH;n++) {
      if(FFT_UINT_DATA[n]>FFT_UINT_DATA_MAX)
      {

FFT_UINT_DATA_MAX=FFT_UINT_DATA[n];
              FFT_Index_MAX=n;
      }
    }
```

```
  // komplette Kurve
  for(n=1;n<SCALE_W;n++) {
    if(Menu.ch1.visible==0) {

ch1_wert2=oszi_adc2pixel(ADC_DMA_Buffer_C[n*2],
Menu.ch1.faktor);

ch1_wert2+=SCALE_Y_MITTE+SCALE_START_X+Menu.ch1.
position;
      if(ch1_wert2<SCALE_START_X)
ch1_wert2=SCALE_START_X;
      if(ch1_wert2>SCALE_MX_PIXEL)
ch1_wert2=SCALE_MX_PIXEL;

UB_Graphic2D_DrawLineNormal(ch1_wert1,SCALE_STAR
T_Y+n,ch1_wert2,SCALE_START_Y+n+1,ADC_CH1_COL);
      ch1_wert1=ch1_wert2;
    }

    if(Menu.ch2.visible==0) {

ch2_wert2=oszi_adc2pixel(ADC_DMA_Buffer_C[(n*2)+
1], Menu.ch2.faktor);

ch2_wert2+=SCALE_Y_MITTE+SCALE_START_X+Menu.ch2.
position;
      if(ch2_wert2<SCALE_START_X)
ch2_wert2=SCALE_START_X;
      if(ch2_wert2>SCALE_MX_PIXEL)
ch2_wert2=SCALE_MX_PIXEL;

UB_Graphic2D_DrawLineNormal(ch2_wert1,SCALE_STAR
T_Y+n,ch2_wert2,SCALE_START_Y+n+1,ADC_CH2_COL);
      ch2_wert1=ch2_wert2;
    }
  }

  // nur die linke hälfte der FFT zeichnen
  // (die rechte ist das Spiegelbild)
  if(Menu.fft.mode!=0) {
```

```
    for(n=1;n<FFT_VISIBLE_LENGTH;n++) {
      fft_wert2=FFT_UINT_DATA[n];
      fft_wert2+=FFT_START_X;
      if(fft_wert2<SCALE_START_X)
fft_wert2=SCALE_START_X;
      if(fft_wert2>SCALE_MX_PIXEL)
fft_wert2=SCALE_MX_PIXEL;

UB_Graphic2D_DrawLineNormal(fft_wert1,FFT_START_
Y+n,fft_wert2,FFT_START_Y+n+1,FFT_COL);
      fft_wert1=fft_wert2;
    }
  }

  UB_LCD_SetLayer_Back();
}


//-------------------------------------------------
---------------
// Zum umrechnen von adc-Wert in Pixel-Position
//-------------------------------------------------
---------------
int16_t oszi_adc2pixel(uint16_t adc, uint32_t
faktor)
{
  int16_t ret_wert=0;

  switch(faktor) {
    case 0 : ret_wert=adc*FAKTOR_5V;break;
    case 1 : ret_wert=adc*FAKTOR_2V;break;
    case 2 : ret_wert=adc*FAKTOR_1V;break;
    case 3 : ret_wert=adc*FAKTOR_0V5;break;
    case 4 : ret_wert=adc*FAKTOR_0V2;break;
    case 5 : ret_wert=adc*FAKTOR_0V1;break;
  }

  return(ret_wert);
}
```

```
//-----------------------------------------------
----------------
// Kirim data melalui UART
//-----------------------------------------------
----------------
void p_oszi_send_data(void)
{
  uint32_t n;
  uint16_t wert1,wert2;
  char buf[40];
//  extern const SM_Item_t UM_01[];
//  extern const SM_Item_t UM_02[];
  //--------------------------------
  // send Screen as Bitmap
  //--------------------------------
  if(Menu.send.mode==6) {
    p_oszi_send_screen();
    return;
  }

  //--------------------------------
  // send settings
  //--------------------------------
  //p_oszi_send_uart("SETTINGS:");
  /*if((Menu.send.mode==0) ||
(Menu.send.mode==1) || (Menu.send.mode==4) ||
(Menu.send.mode==5)) {

sprintf(buf,"CH1=%s/div",UM_01[Menu.ch1.faktor].
stxt);
    p_oszi_send_uart(buf);
  }
  if((Menu.send.mode==2) || (Menu.send.mode==3)
|| (Menu.send.mode==4) || (Menu.send.mode==5)) {

sprintf(buf,"CH2=%s/div",UM_01[Menu.ch2.faktor].
stxt);
    p_oszi_send_uart(buf);
  }
```

```
sprintf(buf,"Time=%s/div",UM_02[Menu.timebase.va
lue].stxt);
  p_oszi_send_uart(buf);
  p_oszi_send_uart("1div=25");

  sprintf(buf,"count=%d",SCALE_W);
  p_oszi_send_uart(buf);*/

  //--------------------------------
  // send data
  //--------------------------------
  //p_oszi_send_uart("DATA:");
//  if((Menu.send.mode==0) ||
(Menu.send.mode==1)) {
//    p_oszi_send_uart("CH1");
//    for(n=0;n<SCALE_W;n++) {
//      wert1=ADC_DMA_Buffer_C[n*2];
//      sprintf(buf,"%d",wert1);
//      p_oszi_send_uart(buf);
//    }
//  }
//  else if((Menu.send.mode==2) ||
(Menu.send.mode==3)) {
//    p_oszi_send_uart("CH2");
//    for(n=0;n<SCALE_W;n++) {
//      wert2=ADC_DMA_Buffer_C[(n*2)+1];
//      sprintf(buf,"%d",wert2);
//      p_oszi_send_uart(buf);
//    }
//  }
//  //Data ini yang dikirim dengan mode yang
sesuai
//  else if((Menu.send.mode==4) ||
(Menu.send.mode==5)) {
//   // p_oszi_send_uart("CH1,CH2");
//    for(n=0;n<SCALE_W;n++) {
//        wert1=ADC_DMA_Buffer_C[n*2];
//        wert2=ADC_DMA_Buffer_C[(n*2)+1];
//      sprintf(buf,"%d,%d",wert1,wert2);
```

```
//      p_oszi_send_uart(buf);
//    }
//  }
  //--------------------------------
  // send FFT
  //--------------------------------
//  if((Menu.send.mode==1) ||
(Menu.send.mode==3) || (Menu.send.mode==5)) {
//    if(Menu.fft.mode==1) {
//      p_oszi_send_uart("FFT:");
//      p_oszi_send_uart("CH1");
//
sprintf(buf,"count=%d",FFT_VISIBLE_LENGTH);
//      p_oszi_send_uart(buf);
//      for(n=0;n<FFT_VISIBLE_LENGTH;n++) {
//        wert2=FFT_UINT_DATA[n];
//        sprintf(buf,"%d",wert2);
//        p_oszi_send_uart(buf);
//      }
//    }
//    else if(Menu.fft.mode==2) {
//      p_oszi_send_uart("FFT:");
//      p_oszi_send_uart("CH2");
//
sprintf(buf,"count=%d",FFT_VISIBLE_LENGTH);
//      p_oszi_send_uart(buf);
//      for(n=0;n<FFT_VISIBLE_LENGTH;n++) {
//        wert2=FFT_UINT_DATA[n];
//        sprintf(buf,"%d",wert2);
//        p_oszi_send_uart(buf);
//      }
//    }
//  }
//
//  // ambil dma ke RAM
      sprintf (buf, "hasil sampling
ADC[simpan1,simpan2]:");
      p_oszi_send_uart(buf);
      for(n=0;n<256;n++){
        simpan1[n] = ADC_DMA_Buffer_C[n*2];
```

```
        simpan2[n] = ADC_DMA_Buffer_C[(n*2)+1];

sprintf(buf,"%d,%d",simpan1[n],simpan2[n]);
        p_oszi_send_uart(buf);
        }
      // Cross correlation algorithm
      // 1. hitung mean simpan1 dan simpan2
      uint32_t meanSimpan1,meanSimpan2;
      uint16_t index;
      meanSimpan1 = 0;
      meanSimpan2 = 0;
      for(index=0;index<256;index++){
            meanSimpan1 =
meanSimpan1+simpan1[index];
            meanSimpan2 =
meanSimpan2+simpan2[index];
      }
      meanSimpan1 = meanSimpan1/256;
      meanSimpan2 = meanSimpan2/256;
      // 2.start crosscorrelation
      int delay;
      int j_index;
      int maxDelay;
      int maxValue=0;
      /* Calculate the correlation series */
      for (delay=-255;delay<256;delay++) {
         crossCorrelation[delay+255] = 0;
         for (index=0;index<256;index++) {
           j_index = index + delay;
         if (j_index < 0 || j_index >= 255)
            continue;
         else{
             crossCorrelation[delay+255] +=
(simpan1[index] - meanSimpan1) *
(simpan2[j_index] - meanSimpan2);

if(crossCorrelation[delay+255]>maxValue){
                  maxValue =
crossCorrelation[delay+255];
                  maxDelay = delay;
```

```
                }
            }
            }
        }

        // ambil delay
        delta=maxDelay;
        // kirim crossCorrelation
        sprintf (buf, "Hasil Perhitungan cross
correlation:");
        p_oszi_send_uart(buf);
        for(delay=-255;delay<256;delay++){
              sprintf (buf, "%d",
crossCorrelation[delay+255]);
              p_oszi_send_uart(buf);
        }
        // fromula: derajat = arcsin(
delta*time_sampling*V_sound/jarak_antar_mic)
        derajat =
asinf(((float)delta*20.0/1000000.0)*384/0.11);
        derajat = derajat*180/PI;
        // parsing float agar bisa dikirim serial
        char *tmpSign = (derajat < 0) ? "-" : "";
        float tmpVal = (derajat < 0) ? -derajat :
derajat;
        int tmpInt1 = tmpVal;                  //
Get the integer (678).
        float tmpFrac = tmpVal - tmpInt1;      //
Get fraction (0.0123).
        int tmpInt2 = trunc(tmpFrac * 10000);  //
Turn into integer (123).

        // kirim derajat
        sprintf (buf, "Hasil perhitungan derajat =
%s%d.%02d\n", tmpSign, tmpInt1, tmpInt2);
        p_oszi_send_uart(buf);


//p_oszi_send_uart("END.");
}
```

```
//-----------------------------------------------
----------------
// string per UART send
//-----------------------------------------------
----------------
void p_oszi_send_uart(char *ptr)
{
      UB_Uart_SendString(COM1,ptr,CRLF);
}
//-----------------------------------------------
----------------
// Kirim layar sebagai bitmap (* .bmp) melalui
UART
// memakan waktu sekitar 20 detik pada 115200
baud
//-----------------------------------------------
----------------
void p_oszi_send_screen(void)
{
  uint32_t n,adr;
  uint16_t x,y,color;
  uint8_t r,g,b;

  // BMP-Header senden
  for(n=0;n<BMP_HEADER_LEN;n++) {
    UB_Uart_SendByte(COM1,BMP_HEADER[n]);
  }

  // cari buffer yang tepat untuk dikirim
  if(LCD_CurrentLayer==1) {
    adr=LCD_FRAME_BUFFER;
  }
  else {
    adr=LCD_FRAME_BUFFER + LCD_FRAME_OFFSET;
  }

  // Kirim semua data warna
  for(x=0;x<LCD_MAXX;x++) {
    for(y=0;y<LCD_MAXY;y++) {
```

```
      n=y*(LCD_MAXX*2)+(x*2);
      color=*(volatile uint16_t*)(adr+n);
      r=((color&0xF800)>>8);  // 5bit red
      g=((color&0x07E0)>>3);  // 6bit green
      b=((color&0x001F)<<3);  // 5bit blue
      UB_Uart_SendByte(COM1,b);
      UB_Uart_SendByte(COM1,g);
      UB_Uart_SendByte(COM1,r);
    }
  }
}
```

Program deklarasi pin out ADC

```
ADC1d_t ADC1d[] = {
  //NAME ,PORT, PIN, CLOCK, Kanal
{ADC_PA5,GPIOA,GPIO_Pin_5,RCC_AHB1Periph_GPIOA,A
DC_Channel_5},   // ADC an PA5 =    ADC12_IN5
{ADC_PA7,GPIOA,GPIO_Pin_7,RCC_AHB1Periph_GPIOA,A
DC_Channel_7},   // ADC an PA7 =    ADC12_IN7
 };
```

Program Inisialisasi ADC Dual Mode Simultaneous

```
void P_ADC_InitADC(void)
{
  ADC_CommonInitTypeDef ADC_CommonInitStructure;
  ADC_InitTypeDef       ADC_InitStructure;

  // Clock Enable
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1 |
RCC_APB2Periph_ADC2, ENABLE);

  //-------------------------------------
  // ADC-Config (DualMode)
  //-------------------------------------

  ADC_CommonInitStructure.ADC_Mode =
ADC_DualMode_RegSimult;
```

```
  ADC_CommonInitStructure.ADC_Prescaler =
ADC1d_VORTEILER;
  ADC_CommonInitStructure.ADC_DMAAccessMode =
ADC_DMAAccessMode_1;
  ADC_CommonInitStructure.ADC_TwoSamplingDelay =
ADC_TwoSamplingDelay_5Cycles;
  ADC_CommonInit(&ADC_CommonInitStructure);

  //-------------------------------------
  // ADC1 (Master)
  //-------------------------------------

  ADC_InitStructure.ADC_Resolution =
ADC_Resolution_12b;
  ADC_InitStructure.ADC_ScanConvMode = DISABLE;
// nur ein Eintrag in der Regular-List ->
disable
  ADC_InitStructure.ADC_ContinuousConvMode =
DISABLE;
  ADC_InitStructure.ADC_ExternalTrigConv =
ADC_ExternalTrigConv_T2_TRGO;
  ADC_InitStructure.ADC_ExternalTrigConvEdge =
ADC_ExternalTrigConvEdge_Rising; // Master wird
getriggert
  ADC_InitStructure.ADC_DataAlign =
ADC_DataAlign_Right;
  ADC_InitStructure.ADC_NbrOfConversion = 1;
  ADC_Init(ADC1, &ADC_InitStructure);

  ADC_RegularChannelConfig(ADC1,
ADC1d[0].ADC_CH, 1, ADC_SampleTime_3Cycles);

  //-------------------------------------
  // ADC2 (Slave)
  //-------------------------------------

  ADC_InitStructure.ADC_Resolution =
ADC_Resolution_12b;
```

```
  ADC_InitStructure.ADC_ScanConvMode = DISABLE;
// nur ein Eintrag in der Regular-List ->
disable
  ADC_InitStructure.ADC_ContinuousConvMode =
DISABLE;
  ADC_InitStructure.ADC_ExternalTrigConv =
ADC_ExternalTrigConv_T2_TRGO;
  ADC_InitStructure.ADC_ExternalTrigConvEdge =
ADC_ExternalTrigConvEdge_None; // Slave darf
nicht gertriggert werden
  ADC_InitStructure.ADC_DataAlign =
ADC_DataAlign_Right;
  ADC_InitStructure.ADC_NbrOfConversion = 1;
  ADC_Init(ADC2, &ADC_InitStructure);

  ADC_RegularChannelConfig(ADC2,
ADC1d[1].ADC_CH, 1, ADC_SampleTime_3Cycles);
}
```

Program Inisialisasi Mode Double Buffer dan DMA

```
void P_ADC_InitDMA_DoubleBuffer(void)
{
  DMA_InitTypeDef       DMA_InitStructure;

  // Clock Enable
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_DMA2,
ENABLE);

  // DMA-Disable
  DMA_Cmd(ADC1_DMA_STREAM, DISABLE);
  // warten bis DMA-Stream disable

while(DMA_GetCmdStatus(ADC1_DMA_STREAM)==ENABLE)
;
  DMA_DeInit(ADC1_DMA_STREAM);

  // DMA-Config
```

```
  DMA_InitStructure.DMA_Channel =
ADC1_DMA_CHANNEL;
  DMA_InitStructure.DMA_PeripheralBaseAddr =
(uint32_t)ADC1_CDR_ADDRESS;
  DMA_InitStructure.DMA_Memory0BaseAddr =
(uint32_t)&ADC_DMA_Buffer_A;
  DMA_InitStructure.DMA_DIR =
DMA_DIR_PeripheralToMemory;
  DMA_InitStructure.DMA_BufferSize =
ADC1d_ANZ*ADC_ARRAY_LEN;
  DMA_InitStructure.DMA_PeripheralInc =
DMA_PeripheralInc_Disable;
  DMA_InitStructure.DMA_MemoryInc =
DMA_MemoryInc_Enable;
  DMA_InitStructure.DMA_PeripheralDataSize =
DMA_PeripheralDataSize_HalfWord;
  DMA_InitStructure.DMA_MemoryDataSize =
DMA_MemoryDataSize_HalfWord;
  DMA_InitStructure.DMA_Mode =
DMA_Mode_Circular;
  DMA_InitStructure.DMA_Priority =
DMA_Priority_High;
  DMA_InitStructure.DMA_FIFOMode =
DMA_FIFOMode_Disable;
  DMA_InitStructure.DMA_FIFOThreshold =
DMA_FIFOThreshold_HalfFull;
  DMA_InitStructure.DMA_MemoryBurst =
DMA_MemoryBurst_Single;
  DMA_InitStructure.DMA_PeripheralBurst =
DMA_PeripheralBurst_Single;
  DMA_Init(ADC1_DMA_STREAM, &DMA_InitStructure);

  // Double-Buffer-Mode
  ADC1_DMA_STREAM->CR|=(uint32_t)DMA_SxCR_DBM;
  ADC1_DMA_STREAM-
>M1AR=(uint32_t)&ADC_DMA_Buffer_B;

  // Flags löschen
  DMA_ClearITPendingBit(DMA2_Stream0,
DMA_IT_TCIF0);
```

```
  DMA_ClearITPendingBit(DMA2_Stream0,
DMA_IT_HTIF0);

  // DMA-enable
  DMA_Cmd(ADC1_DMA_STREAM, ENABLE);

  // warten bis DMA-Stream enable

while(DMA_GetCmdStatus(ADC1_DMA_STREAM)==DISABLE
);
}
```

Halaman Ini Sengaja Dikosongkan

BIODATA PENULIS

Penulis buku adalah Yulizar Edo Pratama Cordova, biasa dipanggil Edo atau lebih akrab teman kampus memanggil Yuli. Lahir di Jember Jumat, 1 Juli 1997. Anak pertama dari tiga bersaudara. kuliah di Fakultas Teknologi Elektro Departemen Teknik Elektro Institut Teknologi Sepuluh Nopember. Tergabung dengan Mahasiswa Pecinta Alam Kalptaru Elektro-ITS. Mengkuti bidang elektronika sangat mengasyikkan dimana banyak pengetahuan baru mengenai komponen dan serba serbi duka kulit tersobek tang potong, ketetesan timah panas, kesentuh solder dalam waktu yang lama, komponen short, komponen mahal, komponen kebanting rusak dan tidak ada harapan atau beli lagi. Yang penting tetap satu dan tetaplah jaya!

Email : yulizaredo@gmail.com
HP/WA : 081252495293
Line/Instagram/Twitter : @yulizaredo