| | | | |
|---|---|---|---|
| | Institut Teknologi Bandung<br>Program Studi Teknik Fisika | | |
| Praktikum | TF2207<br>Laboratorium TF II<br>2 SKS | Dosen | Dr. Ir. Eko Mursito Budi, M.T., IPM<br>Dr.Eng. Muhammad Iqbal, S.T., M.T.<br>Ashari Budi Nugraha, S.T., M.T. |

# 3 SIGNAL I/O

## 3.1 LATAR BELAKANG

Untuk menguji suatu sistem fisis, khususnya rangkaian elektronik, ada dua alat yang sering digunakan yaitu:

- pembangkit sinyal (signal generator) untuk mengumpan sinyal tertentu ke sistem
- penjejak sinyal (signal tracer), untuk mengukur sinyal respon dari rangkaian tersebut.

Dalam konteks ini, sinyal adalah sebuah besaran fisis terukur yang berubah terhadap waktu dan membawa informasi terkait sistem asal sinyal tersebut. Terdapat dua kata kunci yang penting pada sinyal, yaitu "besaran fisis" dan "waktu". Keduanya tidak bisa saling dipisahkan sehingga dalam menghasilkan/mengukur sinyal, kita harus mengeluarkan/mencatat besaran fisis maupun waktu secara teliti dan akurat. Kemudian ada kata ketiga yang mungkin terlewat, yakni "berubah". Agar perubahannya bisa terlihat, maka mengukur sinyal harus dilakukan berkali-kali selama selang waktu tertentu. Semakin cepat perubahan sinyalnya, maka pengukuran juga harus makin sering. Agar dapat berfungsi sebagai pembangkit/penjejak sinyal, maka mikro-kontroller perlu memiliki fasilitas analog I/O dan timer. Pada praktikum ini kita akan mendalami pewaktuan di mikrokontroller dan memanfaatkannya untuk pengukuran sinyal.

## 3.2 KOMPETENSI

Kompetensi yang akan dikuasai peserta setelah menyelesaikan praktikum ini dengan baik adalah:

- Mampu menjelaskan sistem waktu di mikro-kontroller, dan memrogramnnya.
- Mampu mengukur waktu eksekusi task pada mikroprosesor.
- Mampu menentukan periode task yang masih aman untuk eksekuri waktu nyata.
- Mempu menjelaskan struktur data linier buffer dan menggunakannya.
- Mampu menjelaskan dan mengaplikasikan algoritma multi-loop periodik.
- Mampu memrogram aplikasi signal generator
- Mampu memrogram aplikasi signal tracer

## 3.3 ALAT & BAHAN

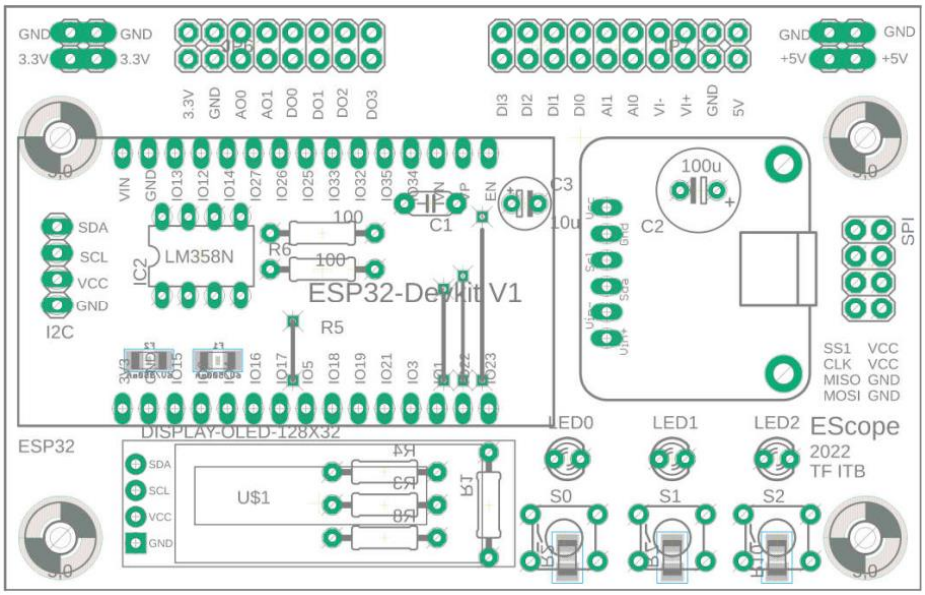
| No | Item | Banyak | Keterangan |
|---|---|---|---|
| 1 | Kit Escope | 1 | Dari Paket kit Lab TF II |
| 2 | Breadboard dan kabel dan komponen paket Lab TF II | 1 | |
| 4 | Kabel USB | 1 | Disiapkan peserta |
| 5 | Komputer (dengan Arduino IDE) | 1 | |

## 3.4 PANDUAN TEKNIS

### 3.4.1 PERANGKAT KERAS

#### 3.4.1.1 ESCOPE

ESCOPE adalah board pengembangan berbasis ESP32 yang dilengkapi dengan beberapa piranti standar, dan didesain agar mudah dikoneksikan ke breadboard.



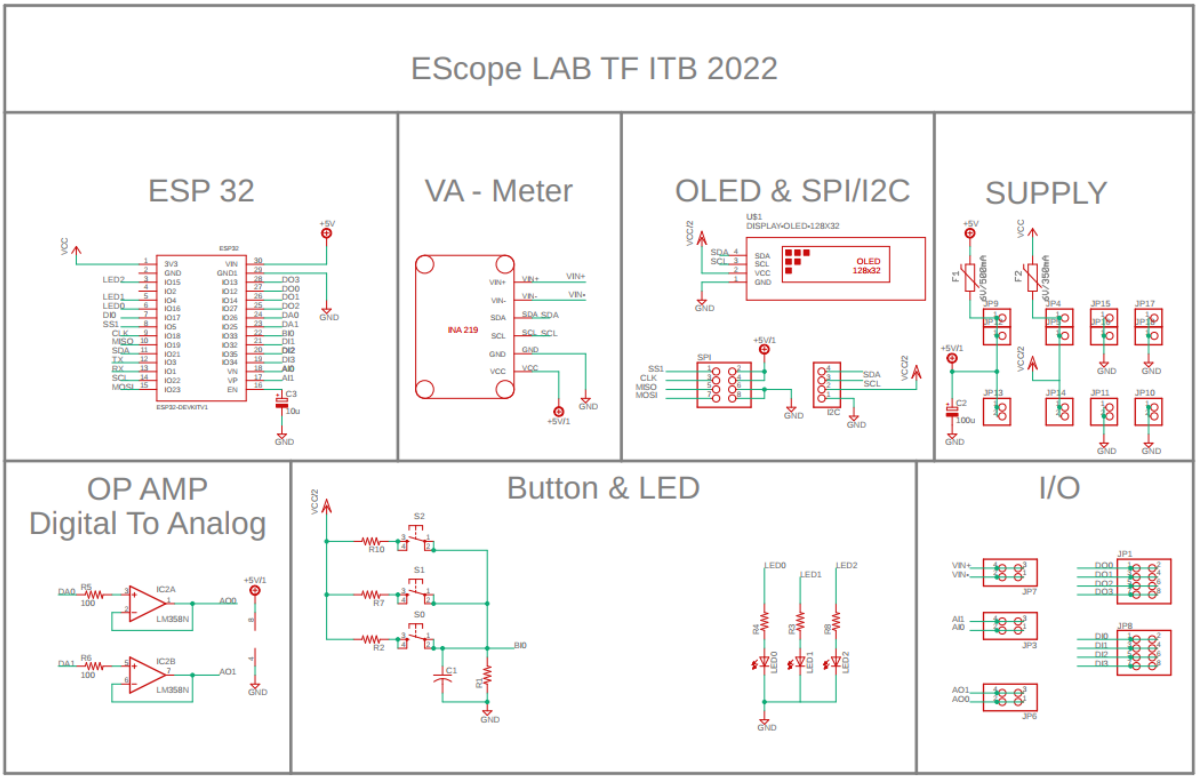

Gambar 3-1 Board ESCOPE

Komponen standar pada ESCOPE meliputi:

- Konektor catu daya 5 Volt, untuk memberi daya ke ESCOPE maupun breadboard
- OLED resolusi 128x32 yang terkoneksi melalui I2C
- INA219 yang juga terkoneksi melalui I2C
- Tiga buat tombol, terkoneksi secara pull-down ke GPIO 13
- Tiga LED (Merah, Kuning, dan Hijau), terkoneksi secara source ke pin GPIO 16, 4, dan 15
- Dua Op-amp sebagai keluaran analog, terkoneksi ke DA pin GPIO 26 dan 25

Selanjutnya terdapat soket AO, DO dan DI untuk sambungan kabel ke breadboard.

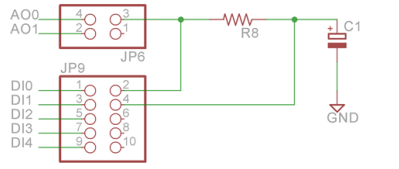
Detail skema ESCOPE adalah sebagai berikut.




**Gambar 3-2 Skema ESCOPE**

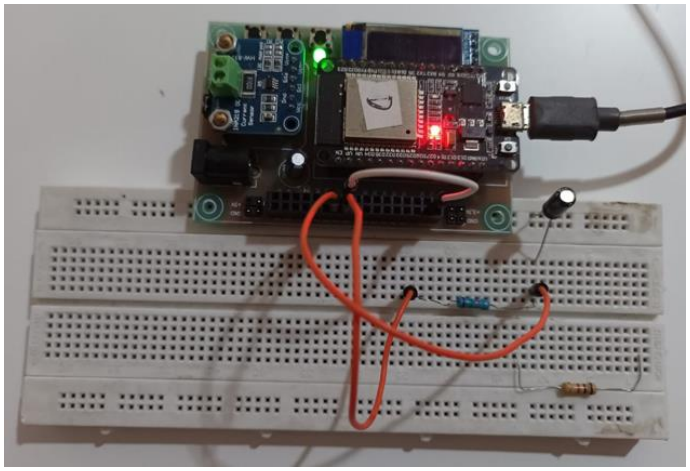
### 3.4.1.2 BREADBOARD

Skema rangkaian dan foto komponen yang perlu dirangkai pada breadboard adalah sebagai berikut.




C1 = 47uF

R8 = 10K

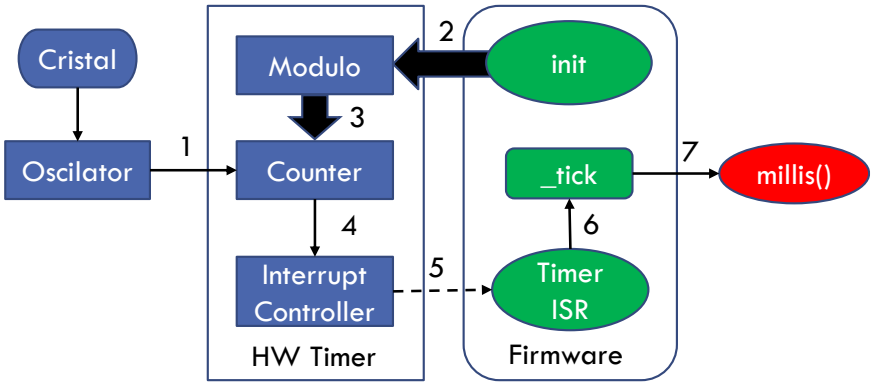


**Gambar 3-3 Skema rangkaian pada breadboard**

### 3.4.1.3 TIMER

Dalam mikro-kontroller, timer adalah perangkat keras yang bertugas menangani waktu. Komponen dan cara kerja umum sistem timer adalah sebagai berikut:

1. Terdapat suatu crystal dan oscilator yang berfungsi membangkitkan pulsa clock dasar. Pada ESP32, clock ini frekuensinya adalah 80MHz.
2. Terdapat register modulo 64 bit, yang perlu diisi angka untuk membagi clock dasar tersebut menjadi frekuensi yang diinginkan. Misalkan diinginkan periode timer 1 mili detik, maka diperlukan frekuensi 1 KHz. Untuk itu modulo harus di-inisialisasi dengan angka 80000.
3. Isi modulo akan dimasukkan ke suatu counter 64 bit.
4. Setiap kali counter menerima pulsa clock, maka isi counter akan berkurang satu. Ketika isi sudah menjadi 0, maka counter akan mengirim pulsa ke interrupt controller, sambil memuat kembali nilai awal dari modulo.
5. Terpicu oleh pulsa dari counter, interrupt controller akan memanggil bagian software yang disebut timer interrupt service routine (ISR).
6. Timer ISR akan menaikkan data global _tick (unsigned int, 32). Dengan demikian _tick ini mencatat lamanya waktu dari awal
7. Isi variabel _tick bisa dibaca oleh program dengan fungsi yang disediakan, dalam hal ini adalah millis().




Gambar 3-4 Diagram blok Timer

Perlu dicatat bahwa waktu yang ditunjukkan oleh timer ini adalah waktu relatif dari saat mikro-kontroller mulai bekerja. Jika ingin mendapatkan waktu absolut sesuai dunia nyata (ada tanggal, pukul), maka diperlukan komponen yang disebut *real-time clock*.

## 3.4.2 PEMROGRAMAN DASAR

### 3.4.2.1 WAKTU

Dalam sistem Arduino/ESP32, waktu dapat dibaca dengan perintah :

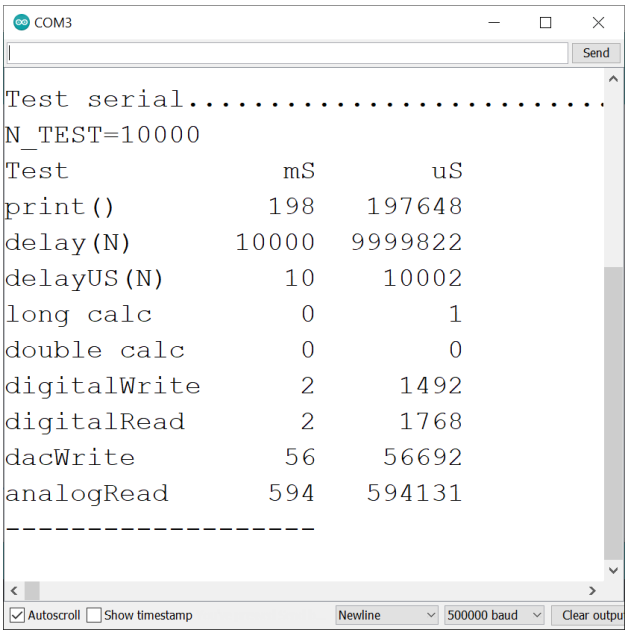
- millis() : membaca waktu dalam mili detik.
- micros() : membaca waktu dalam mikro detik.

Untuk membaca selang waktu, misalnya mengukur lama eksekusi proses, dapat dilakukan dengan contoh berikut:

```
// Test mengukur waktu eksekusi
  us_start=micros();
  for(int i=0; i< N_TEST; i++) {
    task();
```

```
  }
  us_finish=micros();
  tx = (us_finish - us_start) / N_TEST;
```

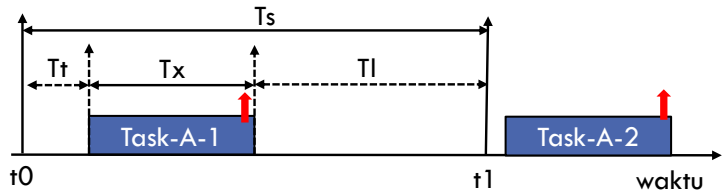
Sebagai contoh, program M3-01 mengukur lama eksekusi berbagai perintah dapat diukur waktu eksekusinya pada program dapat digunakan




Gambar 3-5 Contoh luaran program mengukur waktu eksekusi

### 3.4.2.2 TASK PERIODIK

Dalam istilah computer, task adalah sub program untuk melaksanakan tugas tertentu, biasanya dengan waktu terbatas. Ada task yang berjalan sekali saja, sementara yang lain berjalan berulang kali. Task periodik adalah task yang berulang dengan selang waktu tetap. Dalam hal ini, dikenal beberapa definisi pewaktuan sebagai berikut:




Tx : Waktu eksekusi task
Ts : Periode Task
tn (t0, t1, …) : Trigger time
Tt : Tardiness time
Tl : Slack time
Utilisasi CPU: U = Tx / Ts

Gambar 3-6 Definisi pewaktuan dalam tak periodik

Untuk memudahkan pemrograman task periodik ini, telah disediakan pustaka TFPeriodik. Pada pustaka ini ad akelas **Periodic** untuk mengelola task periodik dengan skala mili detik, dan **PeriodicMicros** untuk skala mikro detik. Contoh pemakaian paling dasarnya adalah sebagai berikut:

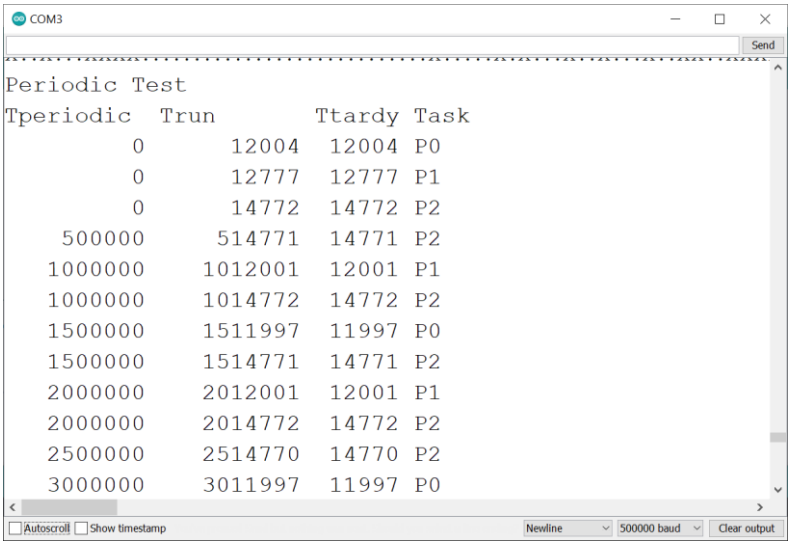
```
// Muat pustakanya
#include <TFPeriodic.h>
```

```
// Pesan obyek, dengan periode yang diinginkan
Periodic p1(1000);

// definisikan task-nya
void taskPeriodik(unsigned long t) {
  Serial.println(t);  // print waktu
}

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

void loop() {
  if (p1.isTime()) {
    taskPeriodik(p1.elapsed());
  }
}
```

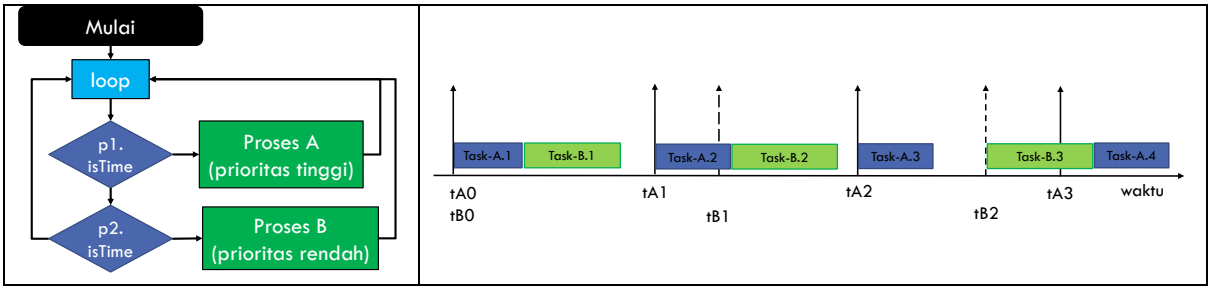
Fungsi-fungsi sediaan kelas Periodic dan PeriodicMicro adalah sebagai berikut

| Fungsi kelas LinierBuffer | Deskripsi |
|---|---|
| reset(unsigned p=0) | Memulai dengan periode tertentu (jika p=0, pakai periode sebelumnya) |
| bool isTime() | Memeriksa apakah waktu periodik sudah sampai |
| void wait() | Menunggu sampai waktu periodik berikutnya sampai |
| unsigned long elapsed() | Mengembalikan waktu dari saat mulai (reset) |
| void setPeriode(unsigned p) | Mengubah periode (waktu mulai tetap) |
| Unsigned getPeriode() | Mengembalikan periode |

Pada kode program M3-02 diberikan contoh lebih kompleks, dimana ada tiga task yang berjalan bersamaan.




**Gambar 3-7 Modul dan skema INA219**




Gambar 3-8 Algoritma multi-loop periodic dan ilustrasi waktu eksekusi task

Perhatikan bahwa algoritma pemrograman memakai kelas Periodic tersebut adalah algoritma multi-loop yang di-trigger oleh waktu. Dalam hal ini, task yang diletakkan lebih awal akan memiliki prioritas lebih tinggi (akan dieksekusi lebih dulu jika memang waktunya sudah tiba). Biasanya, beda prioritas itu diputuskan berdasar pada beberapa pertimbangan, antara lain:

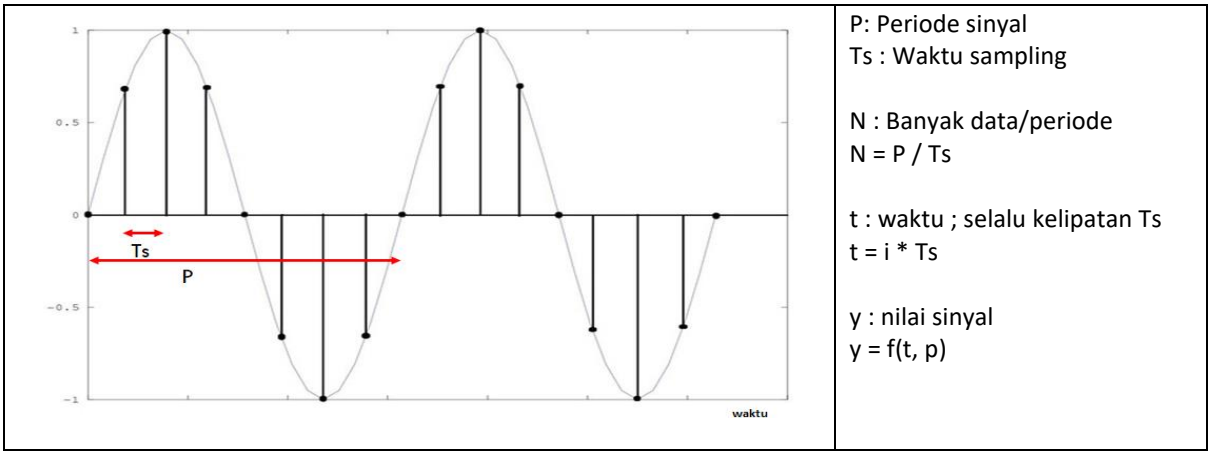
- Pentingnya tugas task tersebut (tak boleh terlambat)
- Periodenya lebih kecil
- Lama eksekusinya lebih kecil

Namun, jika suatu task sudah berjalan, maka eksekusinya tak bisa dihentikan sampai selesai. Mode ini sering disebut sebagai *non-preemptive*. Akibatnya, perhatikan pada ilustrasi, task dengan prioritas lebih rendah bisa saja menunda jalannya task dengan prioritas lebih tinggi. Oleh sebab itu, sangat dianjurkan untuk membuat task yang waktu eksekusinya singkat.

### 3.4.2.3 FUNGSI SINYAL

Sinyal adalah sebuah besaran fisis terukur yang berubah terhadap waktu dan membawa informasi terkait sistem asal sinyal tersebut. Untuk menghasilkan sinyal yang baik, maka mikro-kontroller harus mengeluarkan besaran fisis (tegangan) dengan nilai yang tepat pada waktu yang tepat. Dalam hal ini, mikro-kontroller hanya dapat mengeluarkan sinyal secara periodik, dimana periode ini disebut waktu sampling (bedakan dengan periode sinyal). Dengan demikian sinyal yang dikeluarkan oleh mikro-kontroller:

- bernilai diskrit, karena dikeluarkan oleh DA yang memiliki resolusi tertentu
- waktu diskrit, karena di-sampling dengan periode tertentu

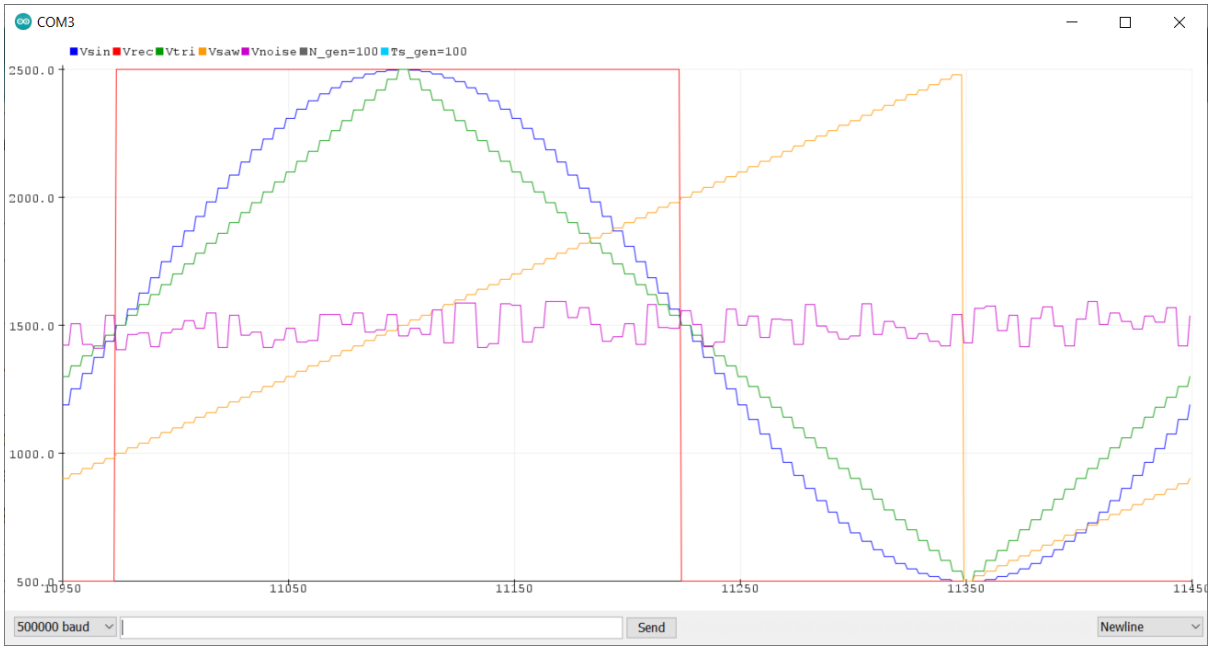



Gambar 3-9 Algoritma multi-loop periodic dan ilustrasi waktu eksekusi task

Untuk menghasilkan sinyal tersebut, maka perlu dibuat fungsi yang menghitung nilai sinyal berdasar waktu dan periode sinyal. Fungsi tersebut kemudian dipanggil oleh task periodik yang berjalan dengan periode sesuai waktu sampling. Berikut ini contoh kode sederhana fungsi sinyal.

```
// rentang sinyal, disesuaikan ke rentang AD (mV) atau rentang DA (0-255)
int signal_max=2500;
int signal_min=500;
int signal_span = signal_max – signal_min;

// fungsi penghasil sinyal
int fsin(unsigned t, unsigned p) {
  y = sin(2 * M_PI * t / p);  // hitung sin rentang -1 .. 1
  return (int) ((y+1)/2*signal_span) + signal_min +0.5; // sesuaikan ke rentang
}

// contoh, panggil fungsi dari task periodik
void taskPeriodik(unsigned long t) {
  Serial.print(t);  // print waktu
  Serial.print(' ');
  Serial.print(fsin(t, periode_signal));  // print nilai sinyal
}
```

Pada program M3-03, terdapat contoh lengkap pembangkit sinyal digital berbagai bentuk yang ditampilkan ke Serial plotter, seperti berikut:

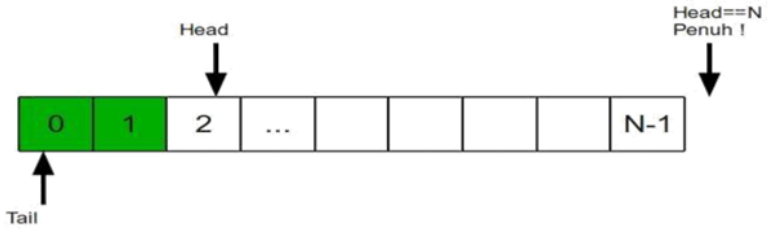



**Gambar 3-10 Contoh berbagai bentuk sinyal digital yang dihasilkan oleh mikro-kontroller**

### 3.4.2.4 LINIER BUFFER

Pengukuran sinyal perlu dilakukan banyak kali untuk selang waktu tertentu. Untuk memudahkan penyimpanan data hasil pengukuran ini, terdapat struktur data yang disebut linier buffer. Pada prinsipnya adalah, linier buffer dapat menyimpan data sebanyak N pada pada suatu array dengan index 0 – N-1. Item data akan disimpan satu persatu mulai dari index 0 hingga penuh sampai index (N-1). Variabel yang menyimpan index

menaruh item ini disebut head. Sementara itu data bisa diambil kembali dari awal hingga habis sesuai index yang disebut tail, sampai habis ketika tail mencapai head.




Gambar 3-11 Ilustrasi struktur data linier buffer

Untuk memudahkan pemrograman, telah disediakan pustaka TFLinearBuffer. Kelas ini dapat menyimpan item berupa array sehingga dapat menyimpan hasil pengukuran beberapa kanal sekaligus. Contoh penggunaannya adalah sebagai berikut:

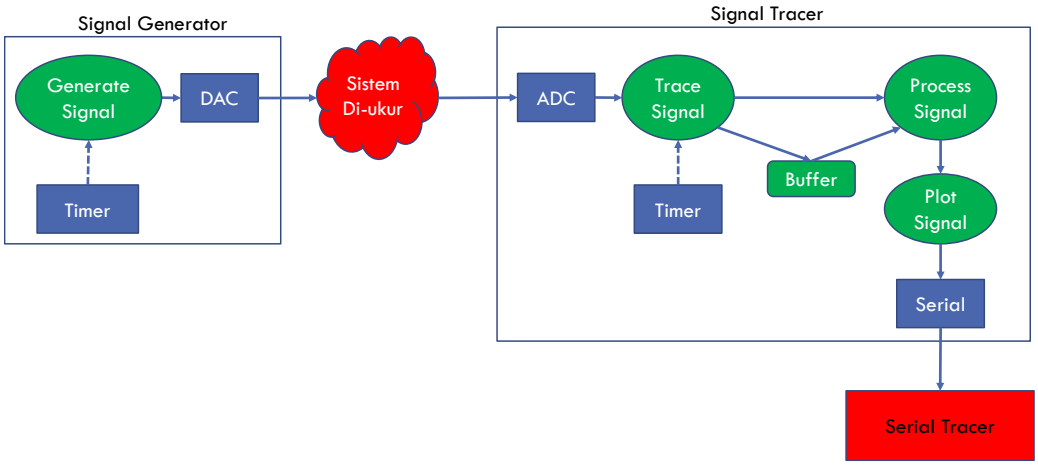
```
#include <TFLinearBuffer.h>

// konstanta
#define N_DATA 500
#define N_SIGNAL 3

LinearBuffer lbuff(N_DATA, N_SIGNAL);

int signal[N_SIGNAL];

// di fungsi bisa dipanggil
lbuff.reset(); // mengosongkan

lbuff.put(signal);   // menaruh satu item ke akhir buffer

lbuff.take(signal);  // mengambil satu item dari awal buffer
```

Beberapa fungsi yang dapat digunakan pada LinearBuffer adalah sebagai berikut:

| Fungsi kelas LinierBuffer | Deskripsi |
|---|---|
| reset() | Mengosongkan buffer |
| void put(int item[]) | Menaruh data ke akhir buffer (head) |
| void take(int item[]) | Mengambil data dari awal buffer (tail) |
| void shiftFirst() | Menggeser data di buffer sehinga data paling lama hilang |
| void shiftLast() | Menggeser data di buffer shg data paling baru hilang |
| bool isFull() | Memeriksa apakah buffer sudah penuh (tak bisa melakukan put) |
| bool isEmpty() | Memeriksa apakah buffer kosong (tak bisa melakukan take) |
| int count() | Melihat banyaknya item di buffer |
| void get(int idx, int item[]) | Mengambil item pada index tertentu (0 = first), tidak mengubah isi buffer |
| int min(int col) | Mengembalikan index item dengan nilai minimal pada kolom tertentu |
| int max(int col) | Mengembalikan index item dengan nilai maksimal pada kolom tertentu |

### 3.4.3 APLIKASI

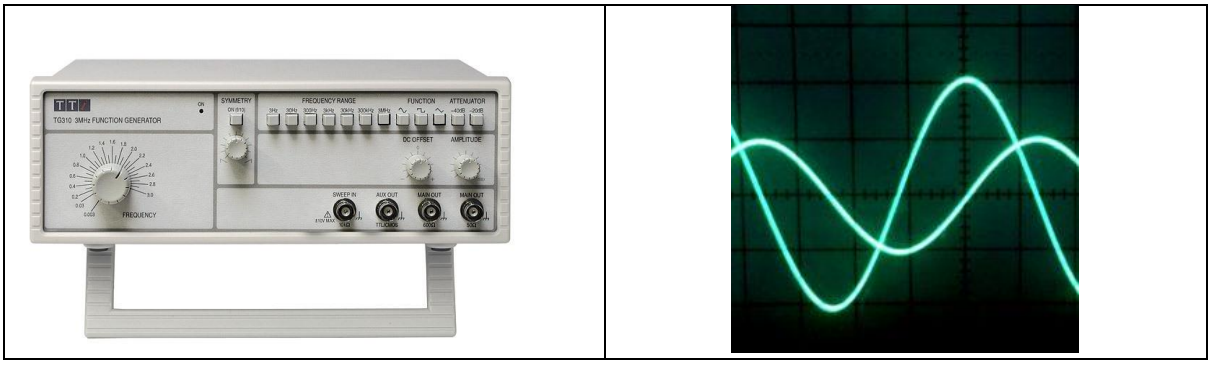
Pada praktikum ini akan dibangun aplikasi untuk pengujian rangkaian elektronik menggunakan signal generator dan signal tracer, dimana diagram blok umumnya nampak sebagai berikut:




**Gambar 3-12 Aplikasi pengujian sistem elektronika**

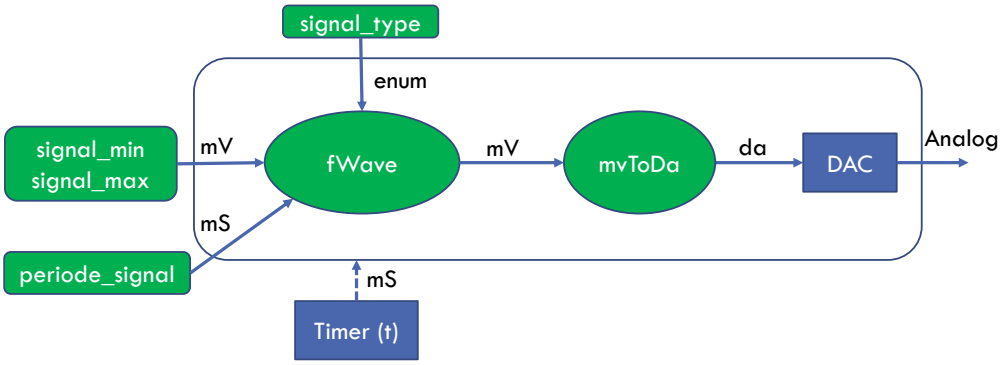
#### 3.4.3.1 PEMBANGKIT SINYAL & PENJEJAL SINYAL ONLINE

Pembangkit sinyal (signal generator) adalah instrument yang berfungsi untuk menghasilkan sinyal listrik, berupa gelombang dengan bentuk, frekuensi, dan amplitude tertentu. Generator sinyal ini dulunya dibuat dari rangkaian elektronik, sehingga sinyal yang dihasilkan adalah analog (kontinyu nilai maupun waktunya) sehingga sangat mulus.



**Gambar 3-13 Analog Signal Generator dan contoh sinyal yang dihasilkannya**

Saat ini, mikro-kontroller dapat dipakai untuk membangkitkan sinyal digital dengan hasil yang lumayan mulus. Diagram aliran datanya cukup sederhana sebagai berikut:




**Gambar 3-14 Diagram aliran data pembangkit sinyal**

Nampak disini bahwa ada fungsi pembangkit sinyal (fWave) yang harus dipanggil secara periodik (berdasar timer). Fungsi ini akan menghitung nilai sinyal sesuai waktu, periode sinyal, rentang, maupun tipe sinyal. Hasilnya kemudiang tinggal dikonversi ke nilai digitalnya untuk dikeluarkan ke DA.

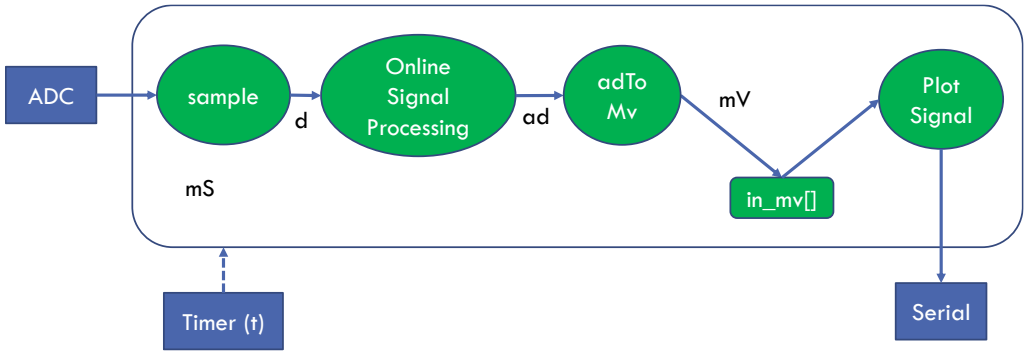
Program M3-04 merupakan contoh pembangkit sinyal. Berikut sebagain potongan kode program intinya:

```
int signal_type = ST_SIN;
unsigned periode_signal = 10000;  // mS
unsigned ts_signal = 100;         // time sampling, ms

Periodic p1(ts_signal);

// task pembangkit sinyal
void generateSignal(unsigned long t) {
  int tp = t % periode_signal;
  out_mv = fWave(signal_type, tp, periode_signal);
  dacWrite(p_out, mvToDa(out_mv));
}

// loop memanggil task pembangkit sinyal secara periodik
void loop() {
  if (p1.isTime()) {
    generateSignal(p1.elapsed());
  }
}
```

Penjejak sinyal (*signal tracer*) adalah instrument untuk mengukur sinyal dan menampilkannya. Dulu, instrument semacam ini lebih dikenal sebagai *oscilloscope* yang hanya dapat menampilkan sinyal berupa gelombang. Hal ini karena oscilloscope dibuat memakai komponen analog yang bekerja berdasar trigger dari sinyal yang diukur. Dengan hadirnya mikro-kontroller, pengukuran tak perlu lagi di-trigger oleh sinyal yang diukur dan tampilannya lebih beragam. Oleh karena itu instrument signal tracer modern sanggup mengukur mixed signal, baik signal periodik (gelombang), aperiodic, maupun digital.

Gambar 3-15 Osiloskop analog dan signal tracer digital (mixed signal)

Pemrograman signal tracer dangan Arduino/ESP32 cukup sederhana jika tampilannya akan dilakukan dengan Serial Tracer. Berikut ini diagram aliran datanya.




Gambar 3-16 Diagram aliran data penjejak sinyal online

Terlihat bahwa ada suatu task yang dijalankan secara periodik untuk melakukan:

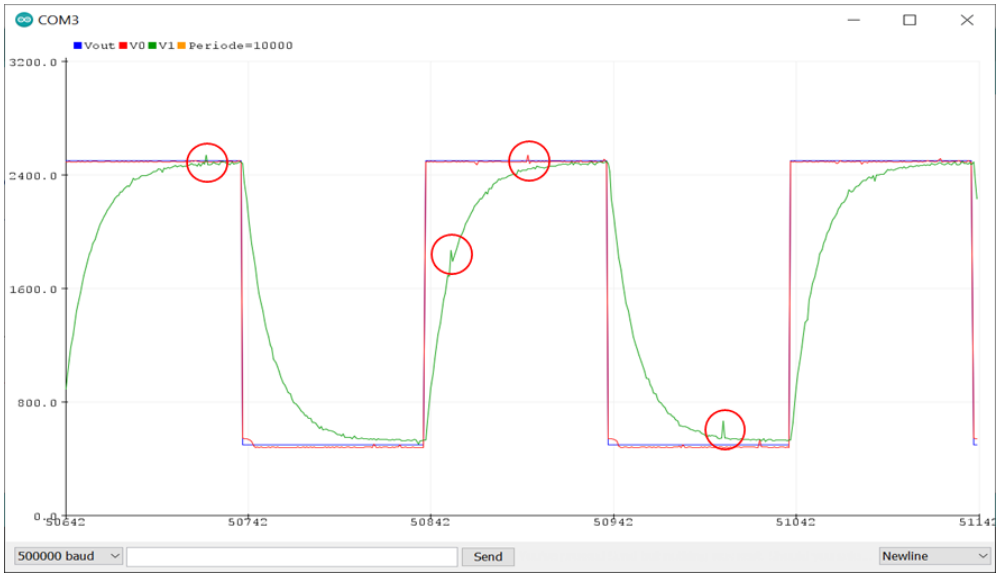
1. Mencuplik tegangan dari ADC
2. Melakukan pengolahan sinyal digital secara online.
3. Mengkonversi hasil pengolahan sinyal ke satuan fisis (mV) dan menyimpannya ke data buffer.
4. Mengirim data melalui komunikasi serial (akan ditampilkan di Serial Plotter).

Dalam hal ini, pengolahan sinyal digital yang dilakukan secara online harus dilakukan sesingkat mungkin. Karena itu hanya operasi esensial seperti menghilangkan noise, jitter, atau spike. Dapat juga dilakukan pengukuran dari ADC beberapa kali, kemudian hasilnya di-rata ratakan. Berikut ini contoh kode penjejak sinyal:

```
#define N_AD 2
int in_mv[N_AD];  // buffer data

int p_inputs[N_AD] = {DI0, DI1};  // pin ADC

unsigned ts_tracer = 20;          // time sampling, ms

Periodic p2(ts_tracer);

// trace signal, tanpa pengolahan sinyal
void traceSignal() {
  for(int i=0; i<N_AD; i++) {
     int ad = analogRead(p_inputs[i]);
     in_mv[i] = adToMv(d);
```

```
  }
}

// plot signal ke serial plotter
void plotSignal() {
   // ...
}

// loop memanggil task penjejak sinyal secara periodik
Periodic p2(1000);
void loop() {
  if (p2.isTime()) {
    traceSignal();
    plotSignal();
  }
}
```

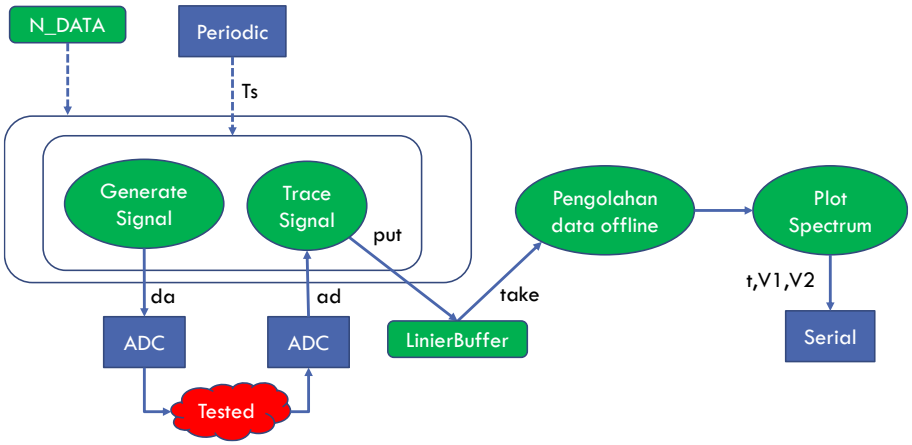
Program M3-04 merupakan contoh signal generator sekaligus signal tracer. Berikut ini contoh tampilannya. Nampak bahwa masih terdapat noise dan spike pada sinyal tersebut, karena tidak dilakukan pengolahan sinyal.




**Gambar 3-17 Contoh tampilan dari program signal_generator + signal_tracer**

### 3.4.3.2 PENGUKURAN TRANSIENT

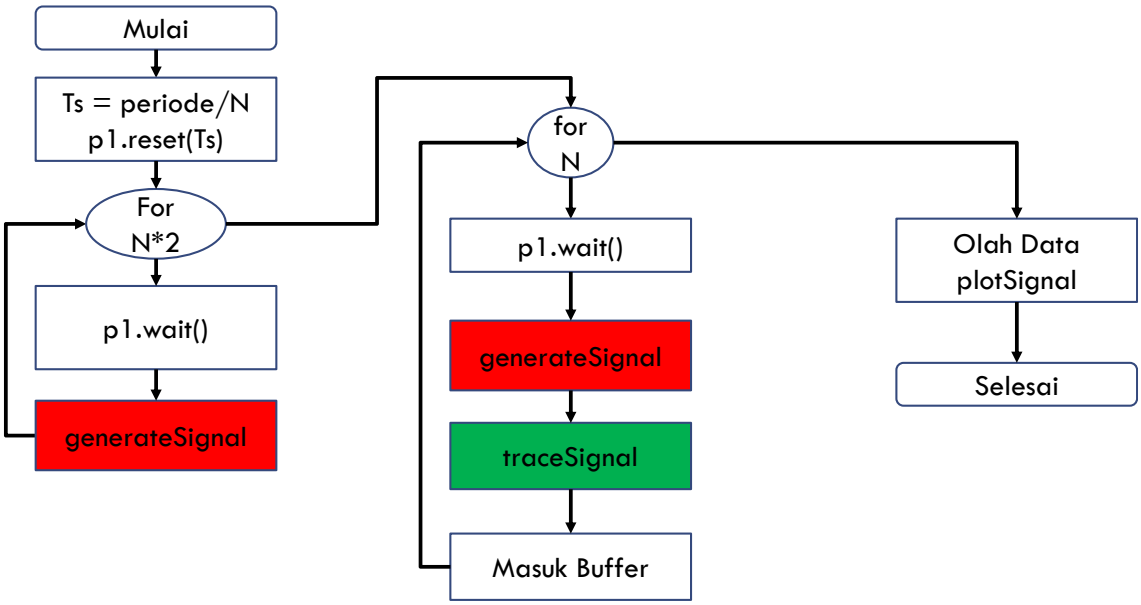
Pengukuran transient adalah salah satu mode pengujian untuk melihat respon transien rangkaian elektronik terhadap masukan sinyal tertentu. Hal ini dapat dilakukan dengan signal generator dan signal tracer yang sudah dijelaskan sebelumnya. Namun, agar dapat melihat respon transien dengan baik maka waktu sampling perlu dibuat sekecil mungkin. Untuk itu perlu digunakan metode tracing secara offline seperti diagram aliran data berikut:




Gambar 3-18 Diagram aliran data penjejak sinyal offline

Nampak disini bahwa operasi penjejak sinyal dibagi menjadi tiga. Bagian terdalam bekerja secara periodik dengan waktu sampling sesingkat-singkatnya; hanya bertugas membangkitkan satu sinyal ke DA, lalu membaca satu data dari AD, lalu langsung menyimpannya ke suatu linier buffer. Proses ini diulang sebanyak N_DATA kali sampai buffer penuh. Setelah itu proses sampling dihentikan, kemudian dilakukan pengolahan data dan pengiriman ke serial. Karena pengukuran sudah selesai, pengolahan data offline dapat melakukan operasi lanjut tanpa batasan waktu yang ketat, misalnya saja mencari waktu tunak, pergeseran fasa, dan lain-lain.

Algoritma pengukuran transien ini nampak pada Gambar berikut. Rahasia dalam pengukuran ini adalah, diperlukan pemanasan dahulu sebelum pengukuran dilakukan. Oleh karena itu perlu diumpankan dulu beberapa gelombang ke rangkaian yang diukur, baru proses penjejakan sinyal dilakukan.




Gambar 3-19 Algoritma pengukuran transien

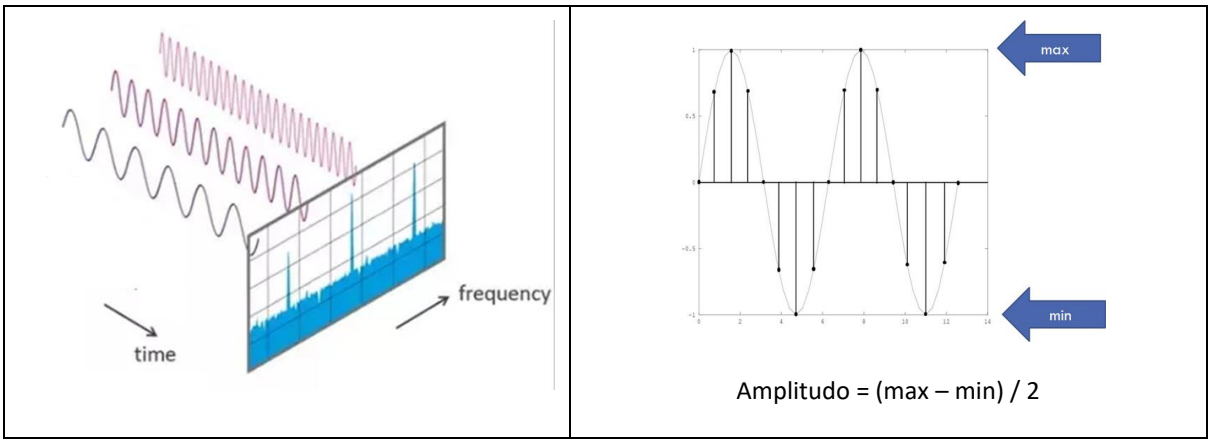
Program lengkap aplikasi ini diberikan pada M3-05. Kode intinya adalah sebagai berikut:

```
#define N_DATA 500
#define N_SIGNAL 2
int in_mv[N_SIGNAL];  // buffer data satu item
LinierBuffer lbuff(N_DATA, N_SIGNAL); // buffer banyak data
```

```
int p_out = AO0;
int p_inputs[N_SIGNAL] = {DI0, DI1};  // pin ADC

int signal_type = ST_RECTANGLE;
unsigned periode_signal = 1000;

unsigned ts_signal = 1;         // time sampling, ms

Periodic p1(ts_signal);

// generate signal
void generateSignal(unsigned long t) {
  int tp = t % periode_signal;
  out_mv = fWave(signal_type, tp, periode_signal);
  dacWrite(p_out, mvToDa(out_mv));
}

// trace signal, masuk buffer
void traceSignal() {
  for(int i=0; i<N_SIGNAL; i++) {
     int ad = analogRead(p_inputs[i]);
     in_mv[i] = adToMv(d);
  }
  lbuff.put(in_mv); // masuk buffer
}

// plot signal ke serial plotter
void plotSignal() {
   lbuff.take(in_mv);  // ambil dari buffer

   // kirim in_mv ke serial
   // ...
}

// fungsi pengukuran transien
void trMeasure() {

  ts_signal = periode_signal / N_DATA;
  p1.reset(ts_signal);

  // pemanasan
  for (int i=0; i<N_DATA*2; i++) {
    p1.wait();
    generateSignal(p1.elapsed());
  }

  for (int i=0; i<N_DATA*2; i++) {
    p1.wait();
    generateSignal(p1.elapsed());
    traceSignal();
  }

  // buffer terisi, tinggal plot
}
```

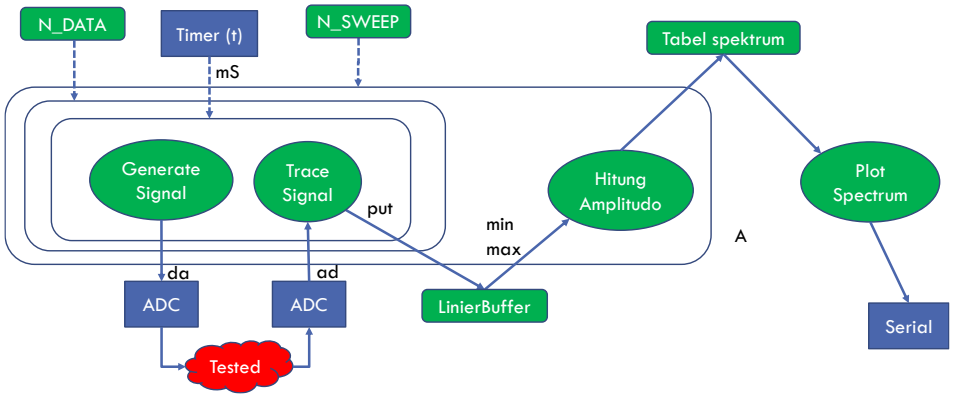
### 3.4.3.3 PENGUKURAN AC-SWEEP

Pengukuran AC-sweep bertujuan untuk melihat respon frekuensi rangkaian elektronik. Untuk itu perlu diumpankan sinyal sinus dari frekuensi rendah ke frekuensi tinggi. Setiap saat dilakukan penjejakan sinyal, kemudian hasilnya diolah sehingga didapat amplitudenya. Output program berupa amplitude ini terhadap frekuensi yang diberikan.




Gambar 3-20 Ilustrasi pengukuran AC-sweep dan proses menentukan amplitudo

Dengan kata lain, AC-sweep adalan pengukuran transien dengan sinyal sinus yang dilakukan beberapa kali (N_SWEEP). Hasil pengukuran ini tersimpan pada linier buffer. Setiap pengukuran transien, akan dihitung amplitudonya dengan fungsi sediaan linier buffer (min dan max), dan disimpan ke tabel spektrum. Setelah terkumpul untuk seluruh frekuensi, tabel spektrum dikirim ke Serial.




Gambar 3-21 Ilustrasi pengukuran AC-sweep dan proses menentukan amplitudo

Kode intinya adalah sebagai berikut:

```
// kode memanfaatkan pengukuran transien

#define N_SWEEP 9

int spectrum[N_SWEEP][2] = {    // tabel spektrum
  {1,0},     // frekuensi, amplitudo
  {2,0}
  ...
};
```

```
// kirim spektrum ke serial
void acPrintSpectrum() {
   // ...
}

// kirim header ke serial
void acPrintHeader() {
   // ...
}

// loop pengukuran AC sweep
void acMeasure () {
  signal_type = ST_SIN;
  for (int i=0; i<N_SWEEP; i++) {

     // hitung periode dari frekuensi di tabel
     int frek = spectrum[i][0];  // ambil frekuensi di tabel
     periode_signal = . . . ;    // hitung periode dr frekuensi

     trMeasure();  // panggil pengukuran transien

     // hasil pengukuran transien ada di buffer

     int a0 = ...; // hitung amplitudo sinyal V0
     int a1 = ...; // hitung amplitude sinyal V1

     spectrum[i][1] = 100 * a1 / a0; // simpan gain dalam persen
  }
}

void acAction() {
  acPrintHeader();
  acMeasure();
  acPrintSpectrum();
}
```

## 3.5 TUGAS AWAL

Siapkan rangkaian praktikum (sesuai 1.4.1.2) pada breadboard, potret bahwa itu sudah siap.

Unduh dan pasanglah pustaka TFPeriodic dan TFLinierBuffer.

Unduh semua kode sumber modul 3.

Sebelum memulai praktikum ini, kerjakan tugas awal berikut:

1. Ambil kode sumber M3-01, untuk mengukur waktu eksekusi berbagai perintah memakai millis() dan micros(). Tangkap luaran program tersebut dari Serial Monitor, salin ke spreadsheet. Hitunglah lama eksekusi setiap perintah. Simpan baik-baik sebagai bahan referensi anda. Lakukan percobaan untuk BAUD 500000 dan 9600 (ubah #define BAUD di awal program).
2. Ambil kode sumber M3-02. Kode itu menjalankan 3 task periodic, dan menganalisis berbagai pewaktuannya. Jalankan dan tangkap luarannya dari Serial Monitor, kira-kira demikian:

```
N-Run    T-Run   U(%) Nr0 Nr1 Nr2    Tt0    Tt1     Tt2     Tx0      Tx1      Tx2
  200  1921611  48.33  97  64  39 1850.8 1643.4    0.0   1663.7   2950.2  14833.6
```

   Jelaskan apa yang dimaksud dengan N-Run (banyak run total), T-Run (Lama run total), U (utilitas), Nr (banyak eksekusi), Tt (tardiness time), dan Tx (execution time).
3. Ambil kode sumber M3-03. Lengkapi kode sumber tersebut untuk dapat menghasilkan gelombang kotak, segitiga, maupun gigi gergaji. Untuk menguji fungsi anda sudah benar, jalankan lalu beri masukan banyak sampling generator (n_gen) = 100 sehingga didapat luaran pada serial plotter seperti pada contoh Gambar 3.10.
4. Pelajari kode M3-04 (metode online). Jelaskan pemrosesan apa yang dilakukan di fungsi traceSignal berikut, dan apa manfaatnya. Ada berapa kali pembacaan AD dalam satu task tersebut ?

```
void traceSignal() {
  long sum[N_AD];
  for (int i=0; i<N_AD; i++) {
    sum[i]=0;
  }
  for (int j=0; j<N_SUM; j++) {
    for (int i=0; i<N_AD; i++) {
      sum[i] += analogRead(p_in[i]);
    }
  }
  for (int i=0; i<N_AD; i++) {
    in_mv[i] = adToMv(sum[i]/N_SUM);
  }
}
```

5. Pelajari kode M3-05 (Transient). Lengkapi tab 30_signal dengan fungsi bentuk sinyal yang lain (dari tugas 4) agar siap melakukan pengukuran transien.
6. Ambil kode sumber pengukuran AC M3-06 yang belum selesai. Coba dengan teman seregu lengkapi kode sumbernya untuk siap melakukan pengukuran AC.

## 3.6 PRAKTIKUM

### 3.6.1 PERIODIC

Pada percobaan ini kita akan menyelidiki pengaruh **lama eksekusi** suatu task terhadap utilitas CPU dan sekaligus tardiness task yang lain. Gunakan kode M3-02. Pada program ini ada 3 task periodik:

- Task-0 : Ts dan Tx tetap, menggunakan pewaktu PeriodicMicros.
- Task-1 : Ts dan Tx tetap, menggunakan pewaktu PeriodicMicros.
- Task-2 : Ts tetap, namun Tx bisa diubah, menggunakan pewaktu Periodic.

Cobalah:

1. Jalankan program tersebut, lalu keluarkan Serial Monitor (ingat, sesuaikan BAUD-nya).
2. Pada layar akan nampak tabel jalannya task (P0 – P2). Setelah selesai akan keluar hasil analisisnya. Tangkap luaran tersebut ke spreadsheet.
3. Ulangi langkah 2 dengan memberi masukan Tx Task-2 dari serial monitor, misal: 1, 5, 10, 15.

Cobalah analisis:

- Catatlah periode (Ts) dan lama eksekusi (Tx) seluruh task dari kode sumber; sementara Tx Task-2 sesuai masukan.
- Apakah pengaruh Tx terhadap utilitas CPU (U) ?
- Apakah banyak eksekusi (Nr0 + Nr1 + Nr2) memang sudah sesuai ?
- Apakah rata-rata lama eksekusi (Tx0 – Tx2) masing-masing task sudah sesuai ?
- Bagaimana pengaruh utilitas terhadap tardiness (Tt0 – Tt2) masing-masing task ?

### 3.6.2 PEMBANGKIT SINYAL

Pada percobaan ini kita akan menyelidiki pengaruh **waktu sampling signal generator** terhadap kehalusan sinyal yang bisa diukur. Untuk itu ambil kode sumber M3-03 yang sudah anda lengkapi fungsi sinyalnya sesuai tugas awal. Pada program ingin ditampilkan satu gelombang penuh pada layar Serial plotter, dimana:

- Layar serial plotter akan menampilkan 500 titik data (N_DATA = 500).
- Periode_signal adalah 1000 mili detik.
- Karena itu signal tracer akan bekerja dengan waktu sampling
  ts_tracer = periode_signal / N_DATA.

Sementara itu, sinyal akan dibangkitkan oleh signal generator yang bekerja dengan waktu sampling berbeda. Untuk memudahkan, yang akan dimasukkan dari layar adalah sample per gelombang (n_generator) sehingga:

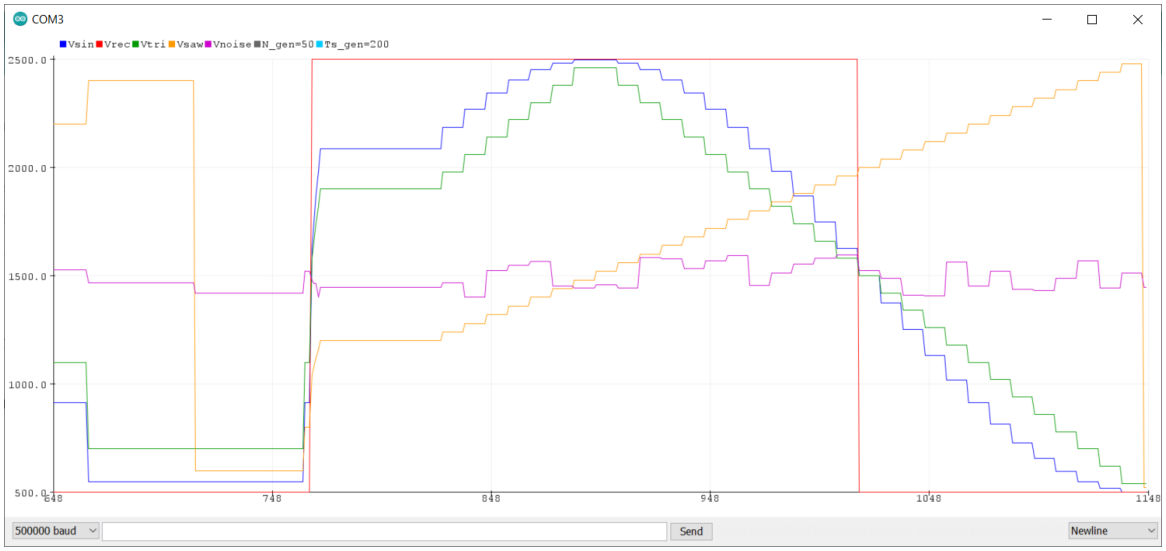
ts_generator = periode_signal / n_generator

Cobalah:

1. Jalankan program tersebut, lalu keluarkan Serial Plotter
2. Mulai dari harga N_gen = 2 sehingga Ts_gen = 5000.
3. Coba capture layar Serial Plotter dengan print-screen. Perhatikan gelombang apa yang sudah terbentuk dengan baik.
4. Ulangi dari Langkah 2, ubah harga N_gen dengan mengetikkan nilai baru di prompt Serial tracer, misalkan 10, 100, 200, 300, 400, 500.

Analisis yang bisa dilakukan :

- Bandingkan sinyal yang dihasilkan, perubahan apa yang terjadi seiring kenaikan N_gen ? Berapa harga N_gen yang menurut anda sudah cukup bagus ?
- Amati gelombang di layar, apakah memang terbentuk 1 gelombang penuh sesuai harapan? Apakah ada perubahan seiring berubahnya waktu sampling Ts_gen ?




**Gambar 3-22 Contoh capture layar setelah mengubah N_gen**

### 3.6.3 PEMBANGKIT SINYAL & PENJEJAK SINYAL ONLINE

Pada percobaan ini kita akan menyelidiki pengaruh **waktu sampling signal tracer** terhadap hasil pengukuran sinyal. Untuk itu telah disediakan kode sumber M3-04. Pelajari kode sumbernya baik-baik, coba pahami:

- Ada sebuah signal generator yang akan menghasilkan sinyal dengan periode tetap 1000 Hz, dengan waktu samplingnya juga tetap, dan sekecil mungkin agar sinyal yang dihasilkan halus. Sinyal ini akan dikeluarkan ke DA, dan masuk ke rangkaian RC.
- Ada sebuah signal tracer yang akan menjejaki dua masukan sinyal dari rangkaian RC tersebut. Hasil penjejakan dikirim ke Serial plotter. Waktu sampling signal tracer ini bisa dipilih dari Serial plotter.
- Pada serial plotter selalu tampil 500 data (N_DATA). Maka jika pewaktuan sistem ini benar, jumlah gelombang yang akan tampil adalah :

  N_gelombang = N_DATA / (periode_signal / ts_tracer).

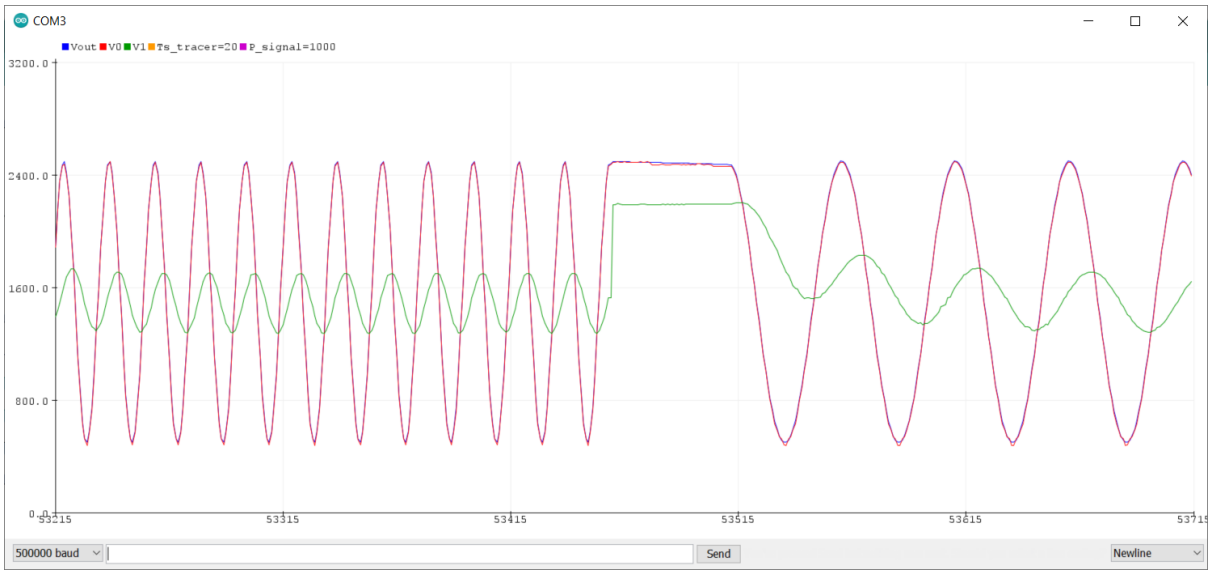
Untuk membuktikannya, lakukan percobaan sebagai berikut:

1. Jalankan program lalu keluarkan layar Serial Plotter.
2. Kirim harga Ts_tracer yang baru, mulai dari 100 ms.
3. Tangkap layar luaran Serial plotter
4. Ulangi dari Langkah 2 dengan harga Ts diturunkan jadi 50, 10, 5, 2, 1.

Analisislah dengan:

- Vout adalah nilai sinyal (mV) sebelum dikeluarkan ke DA, sementara V0 adalah nilai sinyal yang dibaca dari DA dan dikonversi kembali menjadi tagangan (mV). Apa yang terbukti bekerja baik jika keduanya selalu sama ? sebaliknya jika ada beda fasa atau nilai, apa yang salah ?

- Sementara itu hasil V0 dan V1 selalu terjadi beda fasa dan juga nilai. Apakah ini memang sesuai dengan rangkaian RC yang diukur ? jelaskan.
- Coba hitung banyak gelombang pada layar untuk setiap perubahan ts_tracer. Apakah sesuai dengan teori ? Jelaskan jika tak sesuai.




**Gambar 3-23 Contoh percobaan signal tracer berubah waktu sampling**

Untuk analisis lebih dalam, coba pelajari kode sumber signal tracer. Perhatikan bahwa pada setiap waktu sampling, signal tracer menjalankan kode berikut:

```
else if (p2.isTime()) {
    traceSignal();
    plotSignal();
}
```

Sehingga total waktu eksekusinya ditentukan oleh fungsi traceSignal dan plotSignal sebagai berikut:

```
// Baca signal dengan rata-rata
void traceSignal() {
  long sum[N_AD];
  for (int i=0; i<N_AD; i++) {
    sum[i]=0;
  }
  for (int j=0; j<N_SUM; j++) {
    for (int i=0; i<N_AD; i++) {
      sum[i] += analogRead(p_in[i]);
    }
  }
  for (int i=0; i<N_AD; i++) {
    in_mv[i] = adToMv(sum[i]/N_SUM);
  }
}
```

```
void plotSignal() {
  char buff[40];
  sprintf(buff, "%4d %4d %4d",
    out_mv, in_mv[0], in_mv[1]);
  Serial.println(buff);
}
```

Memakai data waktu eksekusi yang sudah anda ukur di tugas awal, cobalah perkirakan waktu eksekusi task signal tracer tersebut. Apakah waktu sampling 1 mili detik masih akan bisa dipenuhi ?

Jika tidak, apa yang bisa dilakukan agar waktu sampling bisa terpenuhi ?

Sebaliknya, jika memang waktu sampling masih bisa diperkecil, apa yang bisa diperbaiki di program ?

### 3.6.4 PENGUKURAN TRANSIENT

Jika algoritma signal generator dan signal tracer sudah benar, maka kita bisa memakainya untuk pengukuran yang sahih. Pertama-tama, akan dilakukan pengukuran transien. Untuk pengukuran ini, dibuat program yang:

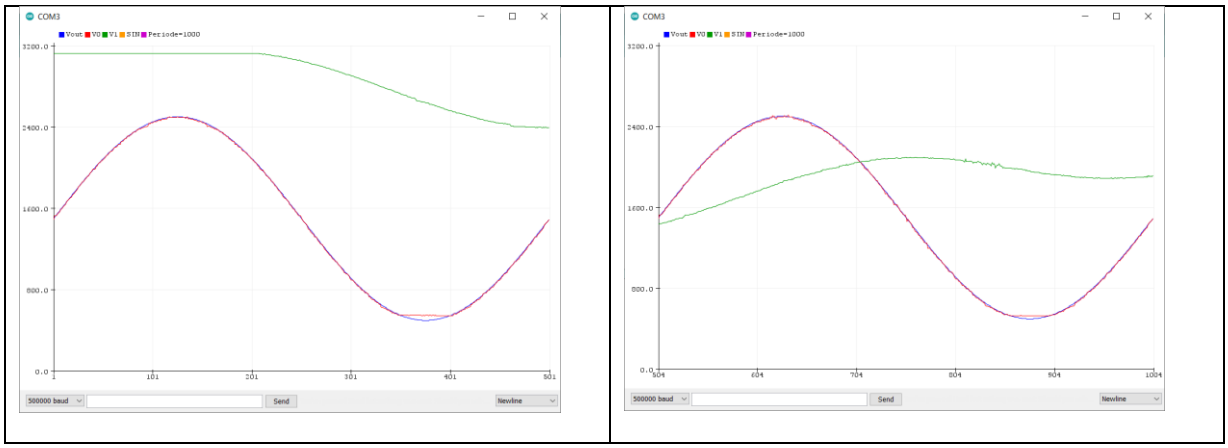
- Menerima masukan satu baris perintah dari Serial Plotter dalam format :
  [bentuk_sinyal] [periode]
- Sistem akan mengeluarkan sinyal ke rangkaian melalui DA, lalu membaca respond-nya dari AD sampai selesai satu (atau lebih) gelombang.
- Sistem mengirim hasil pengukuran ke Serial plotter.

Kode sumber program ini ada di M3-05. Coba muat dan pelajari pada tab-tabnya sebagai berikut:

- M3-05 : awalan
- 10_escope : fungsi dasar escope, sama dengan sebelumnya
- 30_signal : fungsi-fungsi untuk menghasilkan sinyal, HARUS diedit
- 70_data : data global untuk operasi pengukuran (OP, DC, TR, AC).
- 73_TR : bagian operasi pengukuran TR.
- 99_main : program utama

Untuk menjalankan program ini, coba edit tab signal, lengkapi fungsi-fungsi bentuk sinyal yang diperlukan. Setelah berhasil di-upload ke ESP32, cobalah:

1. Ketikkan perintah : TS 1000
2. Amati luaran di Serial Tracer, dan capture layarnya. Apakah memang benar keluar sinyal dengan periode 1000 ms.
3. Coba amati di kode sumber 73_TR, N_WARMUP masih 0. Ubahlah menjadi 2 (atau lebih); lalu coba Kembali. Apakah hasil pengukuran menjadi lebih baik ? Jelaskan.




**Gambar 3-24 Contoh percobaan transien dengan N_WARMUP yang berbeda**

Lanjutkan dengan beberapa percobaan berikut:

1. Coba perintah dengan bentuk sinyal lain (TR, TT, TW). Jika perbaikan kode yang anda lakukan di tugas awal sudah benar, seharusnya akan didapat bentuk sinyal yang sesuai.
2. Coba naikkan periode jadi 5000; lalu turunkan jadi 500, 100. Perhatikan apa yang terjadi. Jelaskan mengapa begitu.

3. Cobalah melakukan pengukuran respon sinyal kotak. Cari frekuensi yang tepat sehingga didapat tampilan menyerupai Gambar 3-17. Bisakah anda menentukan time-constant rangkaian RC dari percobaan ini dengan cukup tepat ?

### 3.6.5 PENGUKURAN AC-SWEEP

Untuk pengukuran AC-SWEEP, disediakan kode sumber M3-06 yang belum selesai. Coba selesaikan program tersebut secara beregu agar dapat melakukan pengukuran AC-SWEEP dengan baik. Luaran yang diharapkan pada Serial Monitor (F konstan, Gain akan beda sesuai percobaan):

```
F(Hz)  Gain (%)
1      97
2      96
4      95
8      94
10     92
20     89
40     77
80     67
100    55
```

Gunakan program tersebut untuk mendapatkan spektrum:

- Rangkaian RC (low pass filter)
- Rangkaian CR (balik menjadi high pass filter).

Tangkap hasilnya ke spreadsheet, lalu buat grafiknya.

## 3.7 TUGAS TANTANGAN

Tugas ini cukup kompleks. Ambil program M3-06, coba gabungkan tab program untuk pengukuran OP, DC (modul sebelumnya). Sempurnakan program utamanya agar program dapat menerima perintah berikut:

```
OP [VA]             : uji OP pada tegangan VA
DC [VA] [VB]        : uji DC sweep dari VA hingga VB
TS [VA] [VB] [F1]   : Uji transien sinus pada frek F1, pada tegangan VA-VB
TR [VA] [VB] [F1]   : Uji transien sinyal rectangle
TT [VA] [VB] [F1]   : Uji transien sinyal triangle
TW [VA] [VB] [F1]   : Uji transien sinyal saw tooth
AC                  : uji AC dengan fekuensi default 1 – 100 Hz
```

Coba gotong royong sebagai satu regu (3 orang) untuk melakukannya (misalkan 1 orang 2 fungsi). Jika anda bisa melakukannya, HEBAT !!!. Anda sudah berhasil memprogram instrument digital sendiri seperti yang digunakan di Lab TF 1.

## 3.8 PUSTAKA

- ADC : https://docs.espressif.com/projects/esp-idf/en/release-v4.2/esp32s2/api-reference/peripherals/adc.html
- ADC: https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/
- Arduino timing : https://www.programmingelectronics.com/millis-arduino/