Pengembangan Program *Interpreter G-code* dan *Motion Control* untuk Mesin CNC *Milling* 3 *Axis* Tipe VMC-100

M. Rizal Ardiansyah[1], Undiana Bambang[2]

[1]*Jurusan Teknik Mesin,Politeknik Negeri Bandung,Bandung 40012*
E-mail : muhammad.rizal.tpkm16@polban.ac.id
[2]*Jurusan Teknik Mesin,Politeknik Negeri Bandung,Bandung 40012*
E-mail : Undiana.bambang@polban.ac.id

ABSTRAK

Politeknik Negeri Bandung memiliki beberapa unit mesin CNC *milling* 3 *axis* bertipe VMC-100 yang mengalami kerusakan pada sistem kontrolnya. Perbaikan mesin-mesin tersebut dapat dilakukan namun membutuhkan biaya besar karena teknologi kontrol yang sudah usang dan jarang digunakan. Diperlukan peremajaan sistem kontrol sehingga mesin-mesin tersebut dapat kembali beroperasi dengan masa pakai yang lebih lama dan biaya operasional yang kecil. Peremajaan dilakukan dengan membuat sendiri sistem kontrol baru berbasis mikrokontroler – *PC* yang lebih modern. Sistem kontrol baru yang dibuat terdiri atas dua fungsi utama, yaitu menerjemahkan g-code (*g-code interpreter*) dan menghasilkan/mengendalikan gerakan (*motion control*) yang menggunakan prinsip *electronic gearbox*. Penggunaan sistem kontrol berbasis mikrokontroler – *PC* meningkatkan kemampuan dan fleksibilitas mesin sehingga dapat beradaptasi dengan mudah apabila ada perubahan pada fungsi *G−code*. Penggunaan sistem kontrol berbasis mikrokontroler – *PC* juga berhasil mengurangi ketergantungan Politeknik Negeri Bandung terhadap pihak lain karena sistem kontrol baru yang mudah dipelajari dan dimodifikasi.

Kata Kunci
CNC, G-Code interpreter, Electronic gearbox, Embedded software

.

1. PENDAHULUAN

Politeknik Negeri Bandung memiliki beberapa unit mesin CNC milling 3 axis bertipe VMC-100. Mesin VMC-100 adalah mesin CNC milling era ’90-an buatan EMCO yang menggunakan sistem kontrol Emcotronics TM02. Saat ini mesin tersebut tidak dapat digunakan karena mengalami malafungsi pada sistem kontrolnya dan disimpan di Lab. CNC Politeknik Negeri Bandung untuk menunggu perbaikan (Gambar 1).

Gambar 1. Mesin VMC-100 yang tersimpan di dalam Lab. CNC Polban

Beberapa solusi untuk memperbaiki mesin ini diantaranya yaitu:

- Mengganti modul kontrol yang rusak dengan yang sama persis.
- Mengganti sistem lama dengan modul kontrol siap pakai (*plug and play*) yang lebih modern.
- Membuat sistem kontrol sendiri.

Solusi pertama memiliki kemungkinan berhasil yang paling tinggi, tetapi solusi ini tidak menjamin mesin dapat digunakan dikemudian hari karena suku cadang yang langka. Solusi ini dapat menyebabkan biaya perawatan melambung tinggi. Solusi kedua akan memastikan mesin mendapat tambahan umur pakai yang cukup lama (bergantung pada sistem kontrol baru yang digunakan). Namun, diperlukan berbagai modifikasi dan penyesuaian pada sistem kontrol baru karena mesin VMC-100 menggunakan mekanisme yang berbeda dengan mesin CNC pada umumnya.

Pada akhirnya, diputuskan untuk membuat sendiri sistem kontrol sederhana yang mampu mengendalikan mesin VMC-100. Walaupun tingkat keberhasilan solusi ini rendah, jika berhasil maka solusi ini akan memakan biaya paling sedikit, dan mengurangi ketergantungan terhadap pihak lain karena perbaikan,

perawatan, dan modifikasi dapat dilakukan secara mandiri.

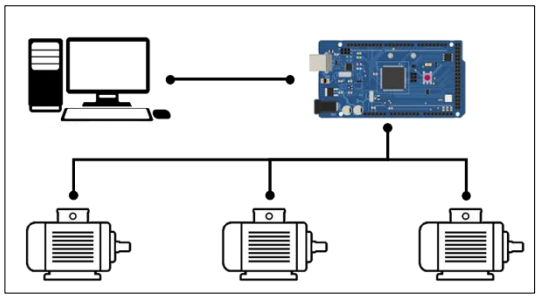
Sebagaimana sistem kontrol eksisting pada mesin VMC-100, sistem kontrol baru harus menggunakan kode-g yang sesuai dengan standar EIA RS-274 atau ISO 6983 sebagai media input. Pada dasarnya kode-g merupakan bahasa pemrograman khusus CNC yang terdiri dari baris kode (*block*) yang juga merupakan gabungan dari kata-kata atau *word*. Sebagai contoh, "G00 X10 Y10" adalah sebuah *block* kode-g dengan tiga *word* dimana "G00" adalah perintah untuk bergerak dengan cepat tanpa melakukan pemakanan, sedangkan X10 dan Y10 adalah koordinat yang dituju [1].

Beberapa percobaan untuk mengganti dan/atau membuat sistem kontrol CNC pernah dilakukan dalam skala yang lebih kecil seperti pada [2], [3], dan [1]. Percobaan-percobaan tersebut telah terbukti berhasil. Oleh karena itu, dibuatlah penelitian ini untuk membuktikan bahwa sistem kontrol buatan sendiri juga dapat berfungsi pada mesin yang lebih besar (kelas industri) seperti VMC-100.

Artikel ini akan membahas tentang metodologi beserta tahapan-tahapan yang digunakan dalam mengembangkan sistem kontrol ini. Pada artikel ini juga akan dibahas secara mendalam tentang cara kerja (algoritma) program penerjemah kode-g (*interpreter*) dan program kendali gerakan (*motion control*) yang digunakan dalam sistem kontrol berbasis mikrokontroler–PC yang telah dibuat.

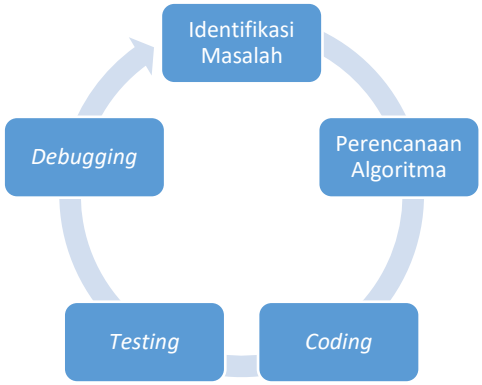
2. METODOLOGI

Sistem kontrol baru akan menggunakan mikrokontroler dan PC (*Personal Computer*) sebagai *hardware*, dimana mikrokontroler akan mengendalikan aktuator berupa motor *servo*. Sedangkan, PC akan memproses masukan dari operator berupa kode-g (*G-code*). PC yang digunakan adalah PC dengan OS Windows, mikrokontroler yang digunakan adalah Atmega 2560 yang berada di *board* Arduino Mega, dan bahasa pemrograman yang digunakan untuk PC dan mikrokontroler adalah C++. Hubungan antar komponen dalam sistem kontrol baru menggunakan topologi atau "bentuk" seperti pada Gambar 2, dimana PC dengan *board* Arduino terhubung melalui komunikasi serial USB sedangkan aktuator terhubung melalui komunikasi paralel dengan Arduino.



Gambar 2. Topologi sistem kontrol baru

Tahapan pembuatan program mengikuti tahapan yang sudah sangat umum seperti dijelaskan pada [4], [5], dan [6]. Tahapan pembuatan terdiri atas identifikasi masalah, perencanaan algoritma, pembuatan kode program, serta pengujian dan perbaikan (*testing and debugging*). Ada pula tahapan dokumentasi program yang tidak akan dibahas pada jurnal ini karena bukan tahapan yang kritis. Secara umum, tahapan ini dapat direpresentasikan seperti pada Gambar 3.



Gambar 3. Tahapan pembuatan program.

2.1 Identifikasi Masalah

Tahap ini diawali dengan melakukan identifikasi fungsi-fungsi yang ada pada sistem kontrol lama dan menentukan fungsi mana yang diperlukan di program kontrol baru. Selanjutnya, dilakukan identifikasi input dan output dari operator mesin untuk membantu menentukan fungsi utama dan fungsi tambahan.

Pada tahap ini juga dilakukan penyortiran kode-g dan kode-m yang diterapkan pada sistem kontrol baru. Penyortiran kode-g dan kode-m dilakukan berdasarkan fungsi tiap kode yang dapat dilihat di [7] dan [1].

2.2 Perencanaan Algoritma dengan *Pseudo-code*

Perancanaan algoritma dilakukan pada setiap fungsi program yang telah diidentifikasi untuk menentukan cara kerja program baru dan menentukan bagaimana

kode program akan ditulis. Proses ini juga dilakukan pada tiap-tiap kode-g yang diimplementasikan.

Proses perencanaan algoritma dapat dilakukan dengan pseudo-code ataupun *flowchart*. Pada jurnal ini, metode *pseudo-code* digunakan dalam membuat program karena dapat menunjukkan struktur program secara lebih jelas dan detil. Penggunaan *pseudo-code* juga dirasa lebih fleksibel dibanding metode *flowchart* karena tidak terikat aturan yang ketat [5].

2.3 Pembuatan Kode Program (*Coding*)

Pada tahap *coding,* semua *pseudo-code* yang telah dibuat di tahap sebelumnya diubah menjadi kode program C++. Program yang dihasilkan dibagi-bagi menjadi beberapa *file* sesuai dengan fungsi program tersebut (*multi-file*) untuk mempermudah pembuatan serta pengujian dan perbaikan. Proses ini menghasilkan sebuah program yang dapat dieksekusi dan digunakan oleh operator mesin CNC.

2.4 Pengujian dan Perbaikan Program (*Testing and Debugging*)

Salah satu metode *testing* yang dilakukan untuk program kontrol ini yaitu dengan mengolah output dari program kontrol yang berupa *text file* (*.txt) menggunakan *software* MS Excel. Hal ini dilakukan untuk mendapatkan grafik yang menggambarkan *tool path* hasil terjemahan kode-g yang akan diikuti oleh mesin CNC.

3. HASIL

Program yang dihasilkan dibagi menjadi enam file kode C++. Keenam file tersebut adalah: Interpreter.cpp, MotionControl.cpp, Serial.cpp, ToolDatabase.cpp, dan WorkOffsetDatabase.cpp. Tiap-tiap file program menangani satu fungsi utama yang dapat dilihat di Tabel 1.

Tabel 1. Fungsi tiap file program

Nama file	Fungsi
Main.cpp	Mengatur jalannya program dan mengolah input dari operator.
Interpreter.cpp	Menerjemahkan kode-g menjadi variabel-variabel gerakan.
* MotionControl.cpp	Mengolah variabel gerakan menjadi pulsa listrik.
ToolDatabase.cpp	Menyimpan dan mengubah data alat potong.
WorkOffsetDatabase.cpp	Menyimpan dan mengubah data benda kerja.
Serial.cpp	Mengatur komunikasi antara PC dengan mikrokontroler.

*) Program berada di mikrokontroler dan juga di PC.

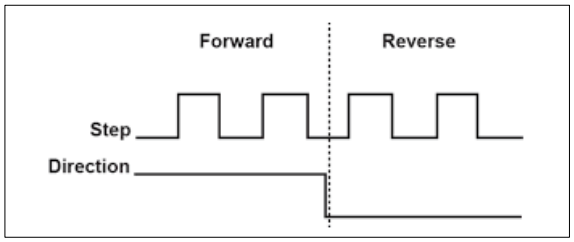
Kode-g dan kode-m yang diimplementasikan pada program kontrol baru merupakan perintah-perintah dasar. Kode-g yang berupa siklus (*canned cycle*) seperti G38.2, G80, G81, dll. tidak diimplementasikan karena akan meningkatkan kerumitan program interpreter. Daftar kode-g dan kode-m yang diimplementasikan dapat dilihat pada Tabel 2.

Tabel 2. Kode-G dan kode-M

Grup Modal	Kode yang diimplementasikan
Grup 1 (Gerakan)	G00, G01, G02, G03
Grup 2 (Bidang kerja)	G17, G18, G19
Grup 3 (Mode jarak)	G90, G91
Grup 5 (*Feed rate*)	G94
Grup 6 (Unit)	G20, G21
Grup 7 (Kompensasi radius)	G40
Grup 8 (*Tool offset*)	G43, G49
Grup 12 (*work offset*)	G54, G55, G56, G57, G58, G59
Grup 0 (non-modal)	G28
Kode-M	M03, M04, M05, M06, M30

3.1 Program Pengatur Gerakan (*Motion Control)*

Program pengatur gerakan (*motion control*) terbagi menjadi dua program, untuk PC dan untuk mikrokontroler. Program *motion control* berfungsi untuk menghasilkan gerakan terkoordinasi/simultan (*coordinated movement*) yang merupakan dasar dari gerakan lurus dan melingkar. Program ini mengolah input dari kode-g berupa koordinat tiga sumbu (X, Y, dan Z) menjadi sinyal listrik *step/pulse and direction* seperti pada Gambar 4 yang akan digunakan untuk mengendalikan motor servo.



Gambar 4. Sinyal *pulse and direction*

Untuk menghasilkan gerakan simultan, program ini terinspirasi oleh metode *counter* dan *increment* [8]. Metode ini bekerja menggunakan prinsip *electronic gearbox* yang menghubungkan poros-poros aktuator secara *virtual*. Metode ini menghilangkan kebutuhan akan *timing* yang akurat karena metode ini menggunakan *counter* untuk menghitung langkah yang telah ditempuh tiap motor servo dan *increment* untuk menentukan kapan motor servo harus melangkah.

Pertama-tama, program menentukan sumbu mana yang harus menempuh jarak terjauh. Sumbu ini kemudian menjadi *master axis* sedangkan sumbu lainnya menjadi *slave axis*. *Master axis* memiliki dua buah *counter* (*step counter* dan *master counter*) dan sebuah nilai *increment* (*master increment*). Tiap *slave axis* hanya memiliki nilai *slave increment* dan satu

slave counter. Nilai awal *slave counter* adalah nilai *slave increment*.

Jika *master axis* melangkah satu kali, maka nilai *step counter* bertambah satu dan *master counter* bertambah sebesar *master increment*. Ketika nilai *master counter* lebih besar dari nilai *slave counter* maka servo *slave axis* akan melangkah satu kali dan nilai *slave counter* akan bertambah sebesar nilai *slave increment*. Program akan terus menggerakkan servo *master axis* sampai nilai pada *step counter* sama dengan jarak yang harus ditempuh *master axis*.

Metode ini kemudian diaplikasikan menjadi sebuah *pseudo-code* yang menjadi algoritma dasar dari program MotionControl.cpp. Pseudo-code untuk program gerak simultan dapat dilihat pada Gambar 5.

```
var X;
var Y;
var slave_axis;
var master_axis;
var master_increment;
var master_counter;
var slave_increment;
var slave_counter;
var step_counter;

MasukanDataXdanY();

X = absolute(X);
Y = absolute(Y);

if X > Y maka master_axis = X dan slave_axis = Y;
if X < Y maka master_axis = Y dan slave_axis = X;

master_increment = KPK(X, Y) / master_axis;
slave_increment  = KPK(X, Y) / slave_axis;

step_counter   = 0;
master_counter = 0;
slave_counter  = slave_increment;

selama (step_counter <= master_axis) adalah benar, lakukan:
{
  Master_Gerak_1_Langkah;
  step_counter = step_counter + 1;
  master_counter = master_counter + master_increment;

  if (master_counter >= slave_counter) maka:
  {
    slave_counter = slave_counter + slave_increment;
    Slave_Gerak_1_Langkah;
  }
}
```

Gambar 5. *Pseudo-code* program gerak simultan

Sebagai contoh, misalkan sumbu X harus bergerak 10 langkah dan sumbu Y bergerak 15 langkah secara simultan. Maka, sumbu Y adalah *master axis* dan X adalah *slave axis*.

Sumbu X akan melangkah jika sumbu Y sudah bergerak 1.5 langkah, sehingga perbandingan jarak sumbu X dan Y menjadi 1:1.5. Perbandingan inilah yang menjadi nilai *master* dan *slave increment.* Namun, karena langkah merupakan jenis variabel *integer*/*discrete* (bilangan bulat), maka nilai perbandingan tersebut perlu disederhanakan menjadi bilangan bulat terlebih dahulu.

Proses penyederhanaan perbandingan pada program dilakukan dengan menggunakan Persamaan 1 dan 2 untuk mendapatkan nilai *slave increment* dan *master increment* secara berturut-turut.

$$slave\ increment = \frac{KPK(master\ axis, slave\ axis)}{slave\ axis} \quad (1)$$

$$master\ increment = \frac{KPK(master\ axis, slave\ axis)}{master\ axis} \quad (2)$$

Kedua persamaan menggunakan perhitungan KPK (kelompok persekutuan terkecil) untuk mendapatkan suatu angka yang dapat dibagi dengan nilai X dan Y. Pada contoh kasus ini, angka tersebut adalah 30. Sehingga, nilai-nilai variabel *increment* menjadi seperti pada Gambar 6.

```
var master_increment = 2;
var master_axis = 15;
var master_counter = 0;
var slave_increment = 3;
var slave_axis = 10;
var slave_counter = 3;
var step_counter = 0;
```

Gambar 6. Nilai-nilai variabel

Perubahan nilai pada variabel *master counter, master increment, slave counter,* dan *slave increment* tiap kali *master axis* melangkah untuk contoh ini dapat dilihat pada Tabel 3.

Tabel 3. Perubahan nilai counter.

Target	: X = 10, Y = 15
Inkremen X	: 3
Inkremen Y	: 2

Posisi Y	Counter Y	Posisi X	Counter X
0	0	0	3
1	2	0	3
2	4	1	6
3	6	2	9
4	8	2	9
5	10	3	12
6	12	4	15
7	14	4	15
8	16	5	18
9	18	6	21
10	20	6	21
11	22	7	24
12	24	8	27
13	26	8	27
Posisi Y	Counter Y	Posisi X	Counter X

14	28	9	30
15	30	10	Irrelevan

3.2 Program Penerjemah Kode-G (*Interpreter)*

Program *interpreter* bekerja dengan membaca program kode-g blok demi blok (baris demi baris) dan menuliskan hasil terjemahannya ke sebuah *text file* (*.txt). Program ini menggunakan fungsi-fungsi *string* dan *fstream* yang ada pada *standard library* C++.

Tahap awal yang dilakukan program ini adalah mengambil satu blok kode-g dan memeriksa apakah ada elemen "komentar" seperti karakter "/", "(", dan ")". Apabila karakter tersebut ditemukan, maka program akan menghapus komentar tersebut.

Selanjutnya, program akan mencari huruf "G" dan huruf "M". Apabila salah satu huruf tersebut ditemukan, program akan mencocokkan angka yang ada setelah huruf tersebut dengan fungsi yang tersedia dan mengaktifkan fungsi tersebut. Jika tidak ditemukan kecocokan, program akan memberi tahu operator secara otomatis dan menghentikan semua proses penerjemahan.

Jika program menemukan huruf selain "G" dan "M", maka huruf tersebut melambangkan parameter untuk fungsi "G" atau "M" yang aktif terakhir. Selanjutnya program akan memasukkan angk-angka yang ada setelah huruf tersebut kedalam variabel tertentu. Sama seperti sebelumnya, program akan berhenti dan memberi pesan kepada operator jika tidak ditemukan kecocokkan.

Setelah seluruh blok diterjemahkan menjadi fungsi dan variabel, program mengeksekusi fungsi-fungsi tersebut dan menuliskan keluaran dari fungsi-fungsi tersebut ke dalam *text file* untuk selanjutnya dibaca oleh program *motion control*.

Fungsi *string* yang digunakan untuk mencocokan adalah *std::string::find()*. Fungsi yang digunakan untuk mendapatkan deret angka setelah huruf perintah adalah *std::string::find_first_not_of()* yang akan memberikan posisi angka terakhir pada deret, dan dengan menggunakan *std::string::substr()* dan posisi angka terakhir maka didapatlah sebuah *substring* yang hanya berisi deret angka tersebut.

Pseudo-code dari kode program *interpreter* dapat dilihat di Gambar 7.

```
1.  file kode_g;
2.  string blok;
3.
4.  ambilBaris(kode_g) simpanKe(blok);
5.
6.  if (ditemukan 'G' pada blok) maka:
7.  {
8.    if (ditemukan "00") aktifkan GerakCepat();
9.    if (ditemukan "01") aktifkan GerakPemakanan();
10.   if (ditemukan "02") aktifkan GerakMelingkarCW();
11.   if (ditemukan "03") aktifkan GerakMelingkarCCW();
12. }
13.
14. else if (ditemukan 'M' pada blok) maka:
15. {
16.   if (ditemukan "03") aktifkan spindleCW();
17.   if (ditemukan "04") aktifkan spindleCCW();
18.   if (ditemukan "05") aktifkan spindleOFF();
19. }
20.
21. else if (ditemukan 'X' pada blok) simpan angka ke variabel X;
22. else if (ditemukan 'Y' pada blok) simpan angka ke variabel Y;
23. else if (ditemukan 'Z' pada blok) simpan angka ke variabel Z;
24.
25. else (tidak ada kecocokan) ERROR! hentikan program;
```

Gambar 7. Pseudo-code program *interpreter*

4. PEMBAHASAN

Penggunaan metode *counter* dan *increment* secara umum sudah dapat bekerja dengan baik. Namun, apabila jarak tempuh sumbu merupakan bilangan prima yang sangat besar seperti pada kasus X = 999961, Y = 999979, Z = 999983, program membutuhkan memori untuk menyimpan nilai *increment* dan *counter* yang cukup besar. Angka-angka tersebut merupakan bilangan prima terbesar yang mendekati volume maksimal mesin yang diasumsikan (1000000 µm x 1000000 µm x 1000000 µm). Pada kasus ini penggunaan variabel tak bertanda 64 bit (uint64_t atau *unsigned long long int*) dapat mengatasi masalah ini karena variabel ini mampu menampung angka hingga 18,446,744,073,709,551,616.

Solusi lain yang jauh lebih baik adalah dengan menggunakan *timer* untuk mengatur kecepatan tiap tiap motor servo. Metode *timer* membutuhkan satu buah *timer counter* untuk tiap motor servo namun metode ini sangat bergantung pada jumlah modul timer pada mikrokontroler sehingga jumlah sumbu yang dapat digunakan sangat terbatas. Metode alternatif lain yang serupa yaitu menggunakan modul DDS (*direct digital synthesizer*) yang dapat menghasilkan berbagai sinyal listrik seperti *square wave*, *sinusoidal wave*, dan *sawtooth wave* pada frekuensi tinggi [9]. Namun, penggunaan DDS akan meningkatkan kerumitan sistem kontrol dan mempersulit proses perawatan yang bertentangan dengan tujuan pembuatan kontrol ini.

Jika dibandingkan dengan sistem kontrol lain seperti GRBL dan NIST RS-274NGC Interpreter, sistem kontrol ini terdiri dari 12 file program yang dipisahkan sesuai fungsi utamanya secara spesifik dan file paling

besar pun tidak lebih dari 900 baris kode (sudah termasuk *comment*) [1] [10].

5. KESIMPULAN

Walaupun dengan keterbatasan volume maksimal mesin, program baru pada sistem kontrol berbasis mikrokontroler-PC terbukti dapat menerjemahkan kode-g dan menggerakkan motor-motor servo untuk gerakan-gerakan 3 *axis* seperti yang diharapkan.

DAFTAR PUSTAKA

[1] T. R. Kramer, F. M. Proctor dan E. Messina, The NIST RS274NGC Interpreter - Version 3, Gaithersburg: National Institute of Standards and Technology., 2000.

[2] A. Quatrano, M. C. D. Simone, Z. B. Rivera dan D. Guida, "Development and Implementation of a Control System for a Retrofitted CNC Machine by Using Arduino," *FME Transactions*, pp. 565-571, 2017.

[3] W. Qin dan P. M. Ferreira, "Design and Analysis of A Small-Scale Cost-Effective CNC Milling Machine.," University of Illinois., Urbana, 2013.

[4] S. Poudel, "Programming Concepts," Science HQ, 18 Februari 2013. [Online]. Available: http://www.sciencehq.com/computing-technology/programming-concepts.html. [Diakses 28 Mei 2020].

[5] J. Valenzuela, "Computer Programming in 4 Steps," International Society for Technology in Education (ISTE), 20 Maret 2018. [Online]. Available: https://www.iste.org/explore/Computer-Science/Computer-programming-in-4-steps. [Diakses 28 Mei 2020].

[6] R. Tyata, A Step in Programming with C, CreateSpace Independent Publishing Platform, 2009.

[7] EMCO Maier G.m.b.H., Instruction Book VMC 100 With Emcotronic TM02, Edition 90-10, EMCO Maier G.m.b.H..

[8] Robin2, "Arduino Forum: Co-ordinated stepper movements using AccelStepper," Arduino, 1 Oktober 2017. [Online]. Available: https://forum.arduino.cc/index.php?topic=503320.0. [Diakses 5 April 2020].

[9] E. Murphy dan C. Slattery, "Direct Digital Synthesis (DDS) Controls Waveforms in Test, Measurement, and Communications," Analog Devices, Inc., Agustus 2005. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/dds-controls-waveforms-in-test.html#. [Diakses 30 Mei 2020].

[10] S. ". K. Jeon dan S. S. Skogsrud, "Github GRBL," Github, 2011. [Online]. Available: https://github.com/gnea/grbl. [Diakses 14 Juni 2020].