PENGARUH PENGGUNAAN BAHASA PEMROGRAMAN MICROPYTHON TERHADAP PENGGUNAAN MEMORI UNTUK IMPELEMENTASI ALOGRITMA PID PADA MIKROKONTROLER STM32

Mukhammad Rifqi Abdillah[1], Santoso[2]
2141220079@student.polinema.ac.id[1], santoso@polinema.ac.id[2]
Politeknik Negeri Malang

ABSTRAK

MicroPython merupakan bahasa pemrograman berbasis interpretasi yang dirancang untuk berjalan pada sistem mikrokontroler dengan sumber daya terbatas. Bahasa ini memberikan kemudahan dalam pengembangan dan fleksibilitas tinggi, namun memunculkan pertanyaan terkait efisiensi penggunaan memori. Penelitian ini bertujuan untuk menganalisis pengaruh penggunaan MicroPython terhadap penggunaan memori dalam implementasi algoritma PID pada mikrokontroler STM32F4. Pengujian dilakukan dengan menjalankan program PID menggunakan platform MicroPython dengan variasi panjang data sebesar 8 hingga 1024 byte. Penggunaan memori diukur secara langsung melalui fungsi monitoring memori internal dan hasilnya dianalisis untuk mengevaluasi kinerja sistem. Hasil pengujian menunjukkan bahwa penggunaan memori meningkat secara signifikan seiring dengan bertambahnya panjang data, dengan penggunaan memori mencapai 10.304 byte pada data berukuran 1024 byte. Temuan ini menunjukkan bahwa manajemen memori otomatis dan overhead interpretasi pada MicroPython menyebabkan penggunaan memori yang lebih besar, yang dapat mempengaruhi efisiensi sistem pada mikrokontroler dengan sumber daya terbatas. Dengan demikian, meskipun MicroPython unggul dari sisi kemudahan pengembangan, penggunaannya dalam implementasi algoritma kendali pada sistem real-time perlu dipertimbangkan secara cermat berdasarkan keterbatasan memori perangkat.

Kata Kunci: MicroPython, PID, STM32.

ABSTRACT

MicroPython is an interpreted programming language designed to run on microcontroller systems with limited resources. This language offers ease of development and high flexibility, yet raises questions regarding memory usage efficiency. This study aims to analyze the impact of using MicroPython on memory usage in the implementation of the PID algorithm on STM32F4 microcontrollers. Testing was conducted by running a PID control program using the MicroPython platform with data lengths varying from 8 to 1024 bytes. Memory usage was measured directly using internal memory monitoring functions, and the results were analyzed to evaluate system performance. The test results showed that memory usage increased significantly as the data length increased, with memory usage reaching 10,304 bytes for data lengths of 1024 bytes. These findings indicate that the automatic memory management and interpretation overhead in MicroPython lead to higher memory usage, which may affect system efficiency on resource-constrained microcontrollers. Therefore, although MicroPython excels in ease of development, its use in implementing control algorithms in real-time systems needs to be carefully considered based on the device's memory limitations.

Keywords: *MicroPython, STM32, PID.*

PENDAHULUAN

Perkembangan sistem kendali berbasis mikrokontroler menuntut efisiensi tinggi dalam penggunaan sumber daya, khususnya memori, untuk memastikan sistem dapat berjalan secara andal dalam kondisi sumber daya terbatas. Salah satu algoritma yang umum digunakan dalam sistem kendali adalah Proportional-Integral-Derivative (PID), yang memiliki sensitivitas terhadap keterbatasan memori pada perangkat keras mikrokontroler. Dalam konteks ini, pemilihan bahasa pemrograman menjadi faktor penting yang memengaruhi efisiensi penggunaan memori sistem. MicroPython merupakan implementasi ringan dari Python 3 yang dirancang khusus untuk perangkat mikrokontroler, menawarkan kemudahan sintaksis, portabilitas, dan kecepatan pengembangan, namun menggunakan pendekatan interpreted yang memiliki kebutuhan memori lebih tinggi dibandingkan dengan bahasa terkompilasi seperti C++. Oleh karena itu, penting untuk mengevaluasi sejauh mana penggunaan MicroPython memengaruhi efisiensi penggunaan memori dalam implementasi algoritma PID pada sistem tertanam(Prabowo & Irwanto, 2023).

Penelitian ini berfokus pada pengujian penggunaan memori algoritma PID yang dijalankan dalam lingkungan bahasa pemrograman MicroPython pada mikrokontroler STM32. Parameter utama yang dianalisis adalah penggunaan memori untuk berbagai panjang data input, dengan tujuan mengetahui dampak nyata dari paradigma pemrograman terhadap efisiensi pemanfaatan sumber daya dalam sistem tertanam. Hasil pengujian diharapkan dapat memberikan referensi empiris dalam memilih bahasa pemrograman yang sesuai untuk aplikasi kendali waktu nyata (real-time control) pada perangkat mikrokontroler dengan keterbatasan sumber daya.

METODE PENELITIAN

Fitur bahasa program MicroPython pada STM32

Dalam pemilihan bahasa pemrograman untuk mikrokontroler, kompatibilitas dengan perangkat keras dan periferal seperti GPIO, UART, SPI, dan I2C menjadi faktor utama. Vendor mikrokontroler biasanya menyediakan pustaka abstraksi perangkat keras (HAL) untuk mempermudah akses ke register dan periferal tertentu. Tanpa pustaka ini, bahasa pemrograman, meskipun memiliki banyak fitur, dapat kurang efektif dalam implementasi sistem kendali berbasis STM32F4, seperti pada ROV (Remotely Operated Vehicle)(ST-Microelectronics, 2024). Selain itu, dukungan fitur bahasa pada perangkat juga berpengaruh, karena beberapa fitur tingkat lanjut mungkin tidak sepenuhnya didukung. Manajemen memori menjadi faktor penting lainnya, dengan tiga metode utama: otomatis, manual, dan garbage collection. Meskipun garbage collection mengurangi kesalahan alokasi memori, proses ini dapat menyebabkan jeda eksekusi yang tidak terduga, yang dapat memengaruhi kestabilan sistem kendali real-time(Bell, 2024).

Pemilihan toolchain dan sistem runtime juga berdampak pada efisiensi dan kinerja sistem. STM32F405RGT6 mendukung beberapa kompiler seperti GCC dan LLVM yang menawarkan berbagai tingkat optimasi ukuran dan kecepatan eksekusi. Alternatif lainnya adalah bahasa terinterpretasi seperti MicroPython, yang mempermudah pengembangan tanpa kompilasi tambahan tetapi memiliki overhead eksekusi lebih tinggi dibandingkan bahasa yang dikompilasi langsung. Selain itu, sistem runtime mengelola stack, heap, threading, dan fitur dinamis lainnya, yang dapat disediakan oleh pustaka standar bahasa atau sistem operasi real-time. Pada STM32, pendekatan bare-metal runtime tanpa sistem operasi juga dapat diterapkan, meskipun dengan fitur yang lebih terbatas dibandingkan sistem yang menggunakan OS penuh(Plauska et al., 2023).

Dalam penelitian ini, MicroPython digunakan sebagai bahasa pemrograman pada STM32F4 untuk mengimplementasikan algoritma PID dengan fokus analisis pada penggunaan memori saat proses eksekusi berlangsung. Konfigurasi sistem dilakukan dengan menggunakan MicroPython port STM32 dengan pustaka machine untuk mengakses GPIO dan UART, serta modul waktu internal untuk pengukuran durasi eksekusi. Lingkungan pengujian menggunakan pendekatan bare-metal tanpa sistem operasi untuk menjaga kesederhanaan dan kontrol penuh terhadap pemanfaatan memori selama pengujian.

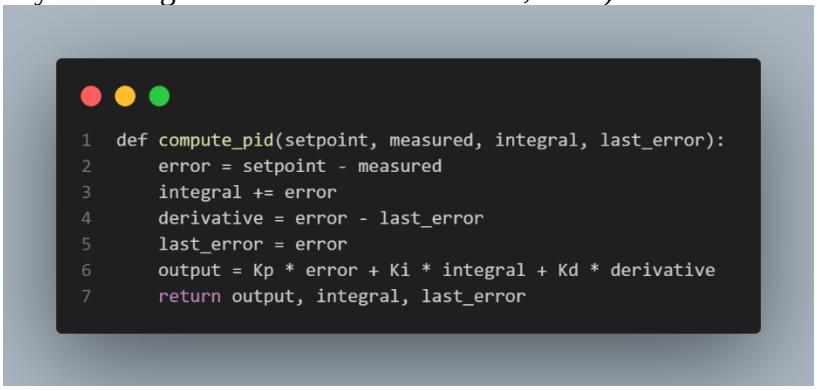
Algoritma yang Digunakan

Dalam penelitian ini, algoritma Proportional-Integral-Derivative (PID) digunakan sebagai metode pengendalian utama dalam sistem kendali permukaan Remotely Operated Vehicle (ROV). Algoritma ini bertugas mengatur aktuator berdasarkan selisih antara nilai setpoint dan nilai aktual dari sensor, dengan tujuan menjaga kestabilan sistem dan menghasilkan respons yang cepat serta presisi(Murdani, 2023). Implementasi PID menjadi komponen penting dalam pengujian performa bahasa pemrograman, karena perhitungannya melibatkan operasi berulang yang sensitif terhadap efisiensi eksekusi pada mikrokontroler STM32F(Solekha & Latifa, 2024).

Performa algoritma ini dievaluasi berdasarkan parameter penggunaan memori sebagai indikator efisiensi sistem. Implementasi PID diuji pada lingkungan MicroPython di mikrokontroler STM32F405RGT6 dengan membandingkan penggunaan memori pada berbagai ukuran data input, yaitu 8, 16, 32, 64, 128, 256, 512, dan 1024 byte. Hasil pengujian dianalisis untuk menilai seberapa efektif MicroPython dalam menangani alokasi dan manajemen memori pada aplikasi sistem kendali real-time berbasis STM32.

Metode Pengujian

Program uji ini ditulis menggunakan MicroPython pada STM32F4 dengan tujuan untuk mengevaluasi penggunaan memori saat implementasi algoritma PID. Program memanfaatkan pustaka machine untuk konfigurasi UART sebagai media pencatatan hasil pengujian, serta pustaka gc untuk memantau penggunaan memori selama eksekusi(Adawiyah Ritonga & Yahfizham Yahfizham, 2023).



```
def compute_pid(setpoint, measured, integral, last_error):
    error = setpoint - measured
    integral += error
    derivative = error - last_error
    last_error = error
    output = Kp * error + Ki * integral + Kd * derivative
    return output, integral, last_error
```

Pada bagian awal, konstanta PID (Kp, Ki, Kd) didefinisikan untuk digunakan dalam fungsi compute_pid, yang menghitung keluaran PID berdasarkan nilai setpoint dan nilai pengukuran simulasi. Algoritma PID dijalankan dengan menghitung error, integral, dan turunan sebagai bagian dari proses kontrol.Data pengujian dihasilkan secara acak dengan panjang data yang bervariasi dari 1 hingga 128 elemen untuk merepresentasikan berbagai kondisi beban data pada sistem kendali. Setiap elemen pada data array diolah menggunakan algoritma PID dengan nilai setpoint acak untuk setiap siklus uji.

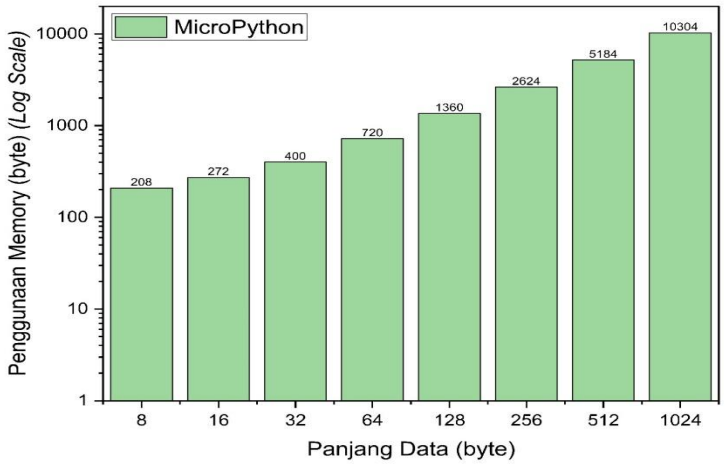
```
for n in [1, 2, 4, 8, 16, 32, 64, 128]:
    data = [random.uniform(0, 200) for _ in range(n)]
    setpoint = random.uniform(0, 200)
    integral = 0
    last_error = 0

    gc.collect()
    mem_before = gc.mem_alloc()
```

Sebelum dan sesudah eksekusi PID, fungsi gc.collect() dipanggil untuk mengoptimalkan penggunaan memori, dan fungsi gc.mem_alloc() digunakan untuk membaca jumlah memori yang teralokasi. Perbedaan nilai memori setelah dan sebelum eksekusi digunakan sebagai data penggunaan memori bersih selama proses PID berjalan.

Hasil pengujian berupa ukuran data (dalam byte), jumlah elemen, dan penggunaan memori dicatat melalui antarmuka UART untuk mempermudah pengambilan data dan analisis lebih lanjut. Dengan pendekatan ini, efisiensi penggunaan memori dalam implementasi PID pada MicroPython dapat dianalisis untuk berbagai ukuran data pada platform STM32, sebagai acuan dalam menentukan kelayakan penggunaan MicroPython pada sistem kendali real-time berbasis mikrokontroler.

HASIL DAN PEMBAHASAN

Pengujian penggunaan memori pada implementasi algoritma PID dengan bahasa pemrograman MicroPython dilakukan dengan variasi panjang data dari 8 hingga 1024 byte. Setiap variasi diuji menggunakan platform STM32F405RGT6 dengan metode pembacaan memori teralokasi sebelum dan sesudah eksekusi menggunakan fungsi gc.mem_alloc(). Perbedaan nilai memori tersebut digunakan sebagai indikator konsumsi memori bersih selama proses eksekusi PID.



Hasil pengujian menunjukkan bahwa terjadi peningkatan penggunaan memori secara signifikan seiring dengan bertambahnya panjang data yang diproses. Pada ukuran data 8 byte, penggunaan memori tercatat sebesar 208 byte. Seiring dengan bertambahnya panjang data, konsumsi memori meningkat tajam hingga mencapai 10.304 byte pada data dengan ukuran 1024 byte. Tren kenaikan ini menunjukkan adanya overhead memori yang

cukup besar pada penggunaan MicroPython, yang kemungkinan berasal dari mekanisme interpreter internal, alokasi objek dinamis, serta proses garbage collection yang berjalan selama eksekusi.

Peningkatan penggunaan memori yang signifikan pada MicroPython memperlihatkan karakteristik bahasa interpretasi yang memiliki kebutuhan memori lebih besar dibandingkan dengan bahasa terkompilasi. Hal ini dapat menjadi pertimbangan penting dalam perancangan sistem kendali berbasis mikrokontroler dengan keterbatasan memori, terutama pada aplikasi sistem kendali real-time yang membutuhkan efisiensi tinggi dalam pemanfaatan sumber daya.

Meskipun demikian, penggunaan MicroPython tetap memiliki keunggulan dari segi kemudahan pengembangan, portabilitas, dan fleksibilitas dalam pembuatan prototipe sistem kendali, yang dapat mempercepat proses pengembangan dan pengujian algoritma seperti PID pada platform mikrokontroler STM32. Oleh karena itu, pemilihan penggunaan MicroPython dalam aplikasi sistem kendali real-time perlu mempertimbangkan keseimbangan antara kemudahan pengembangan dengan keterbatasan memori yang tersedia pada perangkat keras yang digunakan.

KESIMPULAN

Penelitian ini menunjukkan bahwa penggunaan bahasa pemrograman MicroPython pada implementasi algoritma PID di mikrokontroler STM32F4 menyebabkan konsumsi memori yang meningkat signifikan seiring dengan bertambahnya panjang data input yang diproses. Hasil pengujian menunjukkan penggunaan memori mencapai 10.304 byte pada data 1024 byte, menunjukkan adanya overhead memori yang cukup besar akibat mekanisme interpreter dan pengelolaan memori dinamis pada MicroPython. Meskipun demikian, MicroPython tetap menawarkan kemudahan dalam pengembangan dan fleksibilitas tinggi dalam pembuatan prototipe sistem kendali. Oleh karena itu, pemanfaatan MicroPython pada sistem kendali real-time perlu mempertimbangkan keterbatasan memori perangkat untuk memastikan sistem tetap berjalan dengan efisien dan stabil sesuai kebutuhan aplikasi.

DAFTAR PUSAKA

Adawiyah Ritonga, & Yahfizham Yahfizham. (2023). Studi Literatur Perbandingan Bahasa Pemrograman C++ dan Bahasa Pemrograman Python pada Algoritma Pemrograman. Jurnal Teknik Informatika Dan Teknologi Informasi, 3(3), 56–63. https://doi.org/10.55606/jutiti.v3i3.2863

Bell, C. (2024). MicroPython for the Internet of Things. In MicroPython for the Internet of Things. https://doi.org/10.1007/978-1-4842-9861-9

Murdani, M. (2023). Analisis Studi Literatur Penerapan Algoritma Pemrograman pada Internet of Things (IoT). Jurnal Sadewa: Publikasi Ilmu Pendidikan, Pembelajaran Dan Ilmu Sosial, 2(1), 244–255. https://doi.org/10.61132/sadewa.v2i1.507

Plauska, I., Liutkevičius, A., & Janavičiūtė, A. (2023). Performance Evaluation of C/C++, MicroPython, Rust and TinyGo Programming Languages on ESP32 Microcontroller. Electronics (Switzerland), 12(1). https://doi.org/10.3390/electronics12010143

Prabowo, N. K., & Irwanto, I. (2023). The Implementation of Arduino Microcontroller Boards in Science: A Bibliometric Analysis from 2008 to 2022. Journal of Engineering Education Transformations, 37(2), 106–123. https://doi.org/10.16920/jeet/2023/v37i2/23154

Solekha, R., & Latifa, U. (2024). Sistem Kendali Proportional Integral Derivative (PID) Menggunakan Mikrokontroler Arduino Pada Thinkercad. ELECTRON Jurnal Ilmiah Teknik Elektro, 5(1), 89–97. https://doi.org/10.33019/electron.v5i1.108

ST-Microelectronics. (2024). STM32F412xE STM32F412xG. January.