

# CONTROLADOR PID PARA PLANTA DE PROCESSOS CONTÍNUOS

## PLATAFORMA LUATOS ESP32C3 – NÚCLEO RISC-V

### PLATAFORMA DE DESENVOLVIMENTO BIPES

Outubro 2024

---

#### Objetivo

Descrever as etapas necessárias para a implementação de um controlador PID em uma planta de processos contínuos utilizando a plataforma BIPES.

#### Introdução

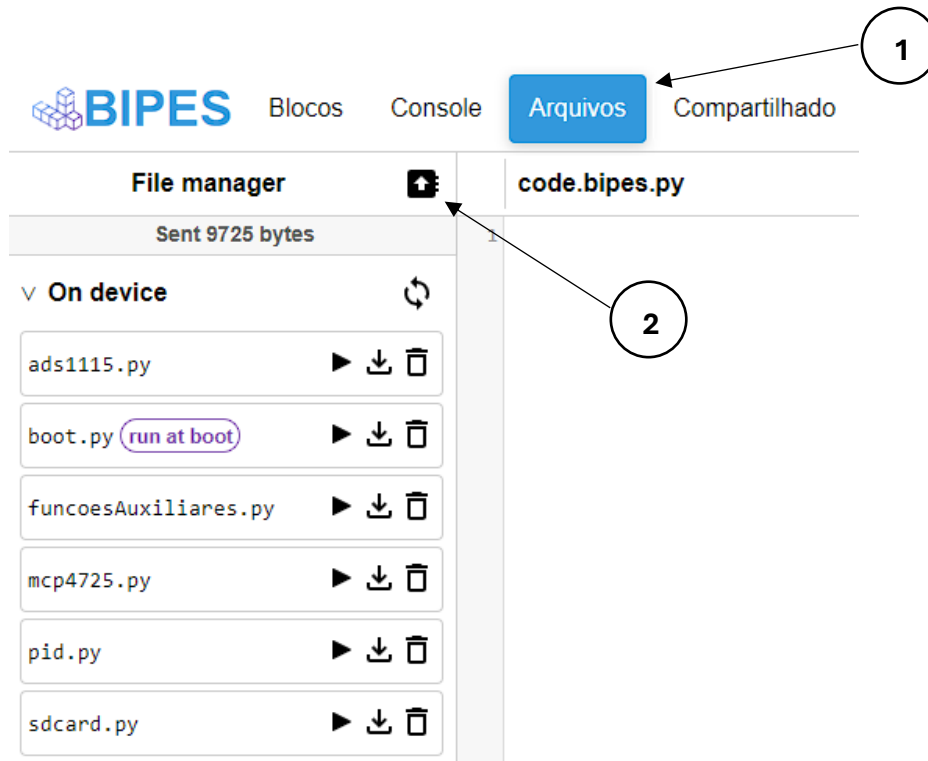
O sistema controlado é composto por um reservatório, um conjunto motobomba e o controlador PID. A variável controlada é o nível do reservatório que pode variar de 0 a 100%, a atuação do controle se dá por um sinal de saída de 0 a 10 V que atua sobre um inversor de frequência, este último, controla a vazão de entrada do tanque acionando um o conjunto motobomba. O *feedback* do sistema é obtido com um transmissor de nível que gera um sinal de 4 a 20 mA, onde 4 mA representa tanque vazio. No cerne do hardware utilizado está um microcontrolador ESP32-C3 com núcleo RISC-V. O esquema elétrico completo pode ser visto no **APÊNDICE 1**, a plataforma utilizada nesta aplicação é a LuatOS, mais informações: [https://templates.blakadder.com/luatos\\_CORE-ESP32.html](https://templates.blakadder.com/luatos_CORE-ESP32.html)

#### Desenvolvimento

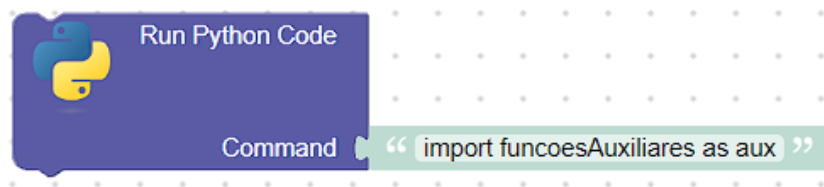
As próximas etapas descrevem o passo-a-passo para implementação do sistema, assume-se que o hardware já tenha o microPython instalado e que a placa já esteja conectada na plataforma Bipes, para mais informações sobre esses pré-requisitos acesse: <https://bipes.net.br/pt/docs.html>

1. Na plataforma Bipes acesse a aba **Arquivos (1)** clique sobre o botão **Upload script to device (2)** e faça o *upload* de todas as bibliotecas necessárias para o funcionamento correto do sistema.

As bibliotecas podem ser baixadas no repositório do projeto: <https://github.com/andreroberto83/PID-ARM-RISCV>



2. Adicione um bloco Run Python Code com o comando: `import funcoesAuxiliares as aux`.



3. Declare as variáveis contadorMinuto, dados, incrementaSp, inicialização, modo, nível, saída, sp, tempo500ms e txt.



4. Faça a inicialização de todas as variáveis.



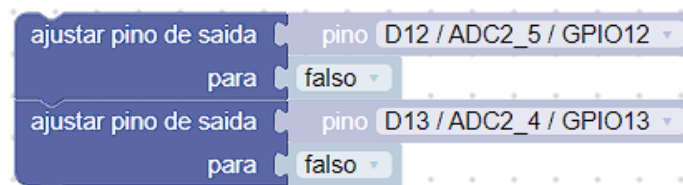
```
definir contadorMinuto para 0
definir dados para falso
definir incrementaSp para verdadeiro
definir inicializacao para falso
definir modo para falso
definir nivel para 0
definir saida para 0
definir sp para 10
definir tempo500ms para 0
definir txt para ""
```

5. Inclua um bloco de interrupção do Timer 0 e as ações que deverão ser executadas na rotina de interrupção do Timer 0.



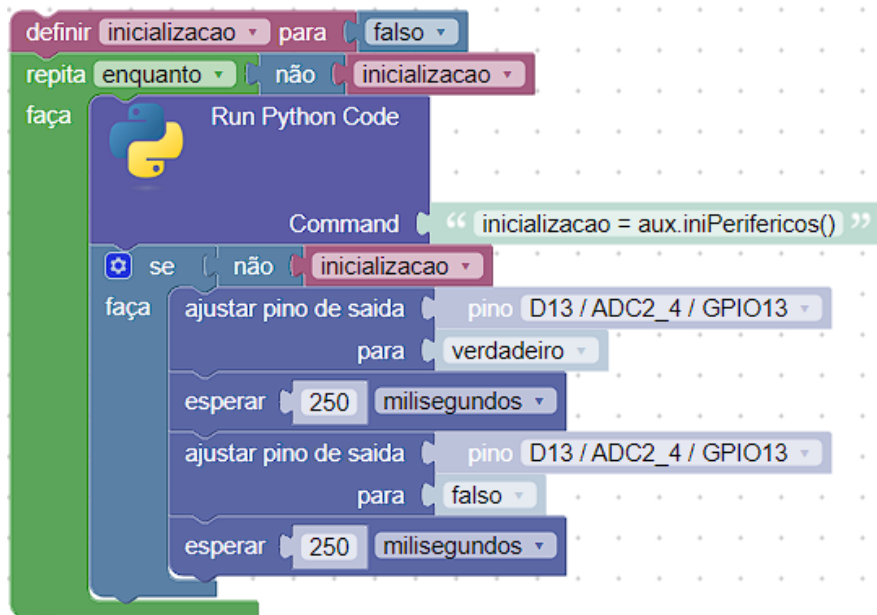
```
Timer # 0 do every 500 ms
  alterar tempo500ms por 1
  definir dados para verdadeiro
```

6. Configure os leds da placa LuatOS.



```
ajustar pino de saida pino D12 / ADC2_5 / GPIO12 para falso
ajustar pino de saida pino D13 / ADC2_4 / GPIO13 para falso
```

7. Inicialize os periféricos

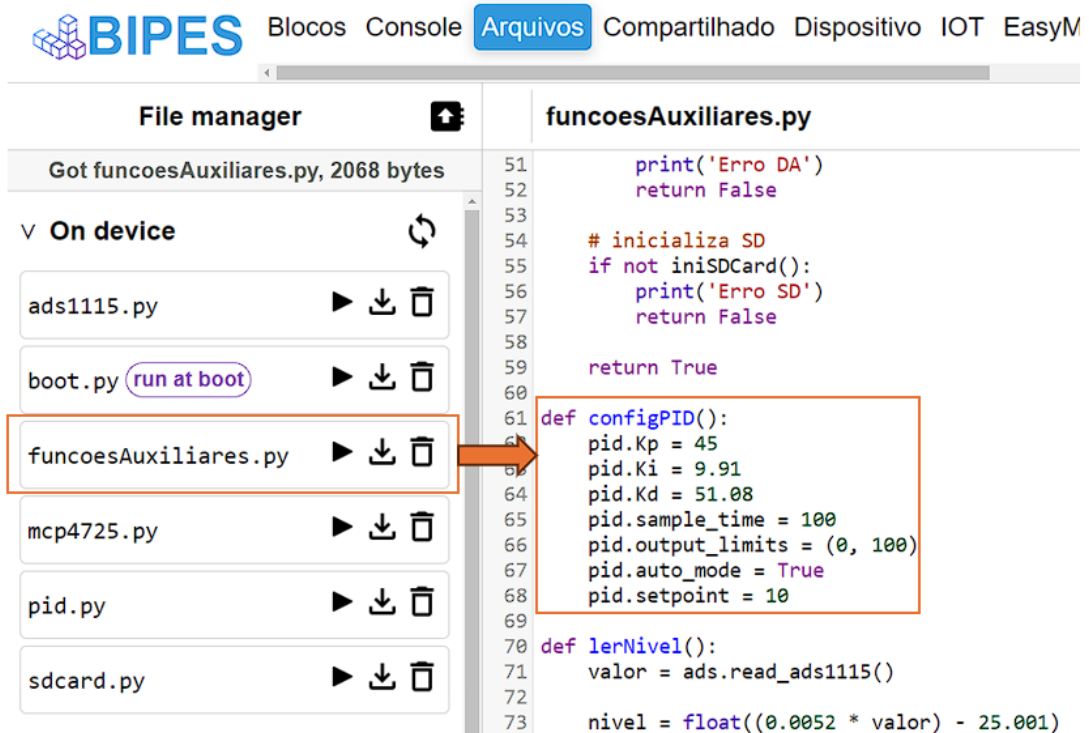


```
definir inicializacao para falso
repita enquanto não inicializacao
  faça
    Run Python Code
    Command " inicializacao = aux.iniPeriféricos() "
    se não inicializacao
      faça
        ajustar pino de saida pino D13 / ADC2_4 / GPIO13 para verdadeiro
        esperar 250 milisegundos
        ajustar pino de saida pino D13 / ADC2_4 / GPIO13 para falso
        esperar 250 milisegundos
```

## 8. Ajuste os parâmetros do controlador PID

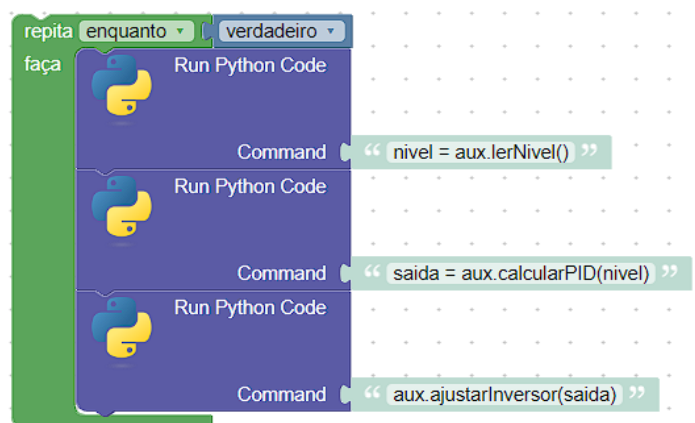


8.1. (Opcional) Caso queira mudar os parâmetros de sintonia Kp, Ki e Kd acesse a aba Arquivos clique sobre a biblioteca `funcoesAuxiliares.py` e altere os parâmetros na função `configPID()`.

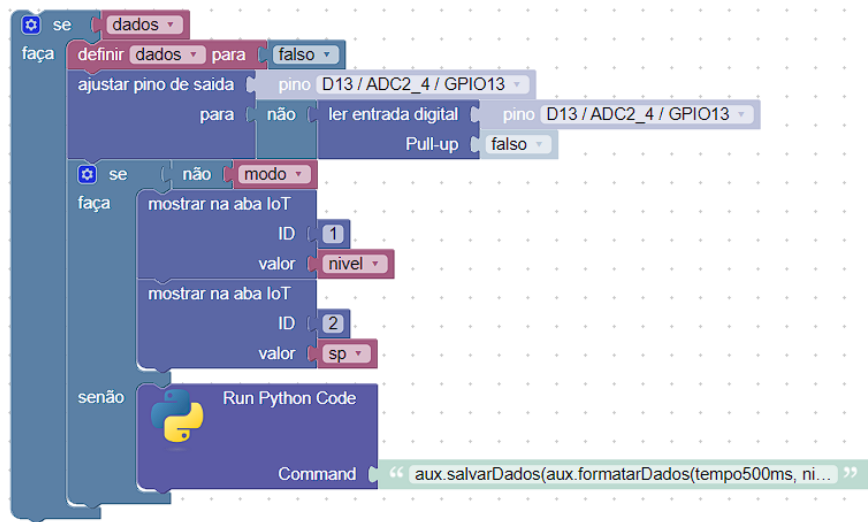


Não esqueça de clicar em **Save** para salvar as alterações ➡  **Save**

## 9. Implemente um laço de repetição infinito e inclua os seguintes blocos:

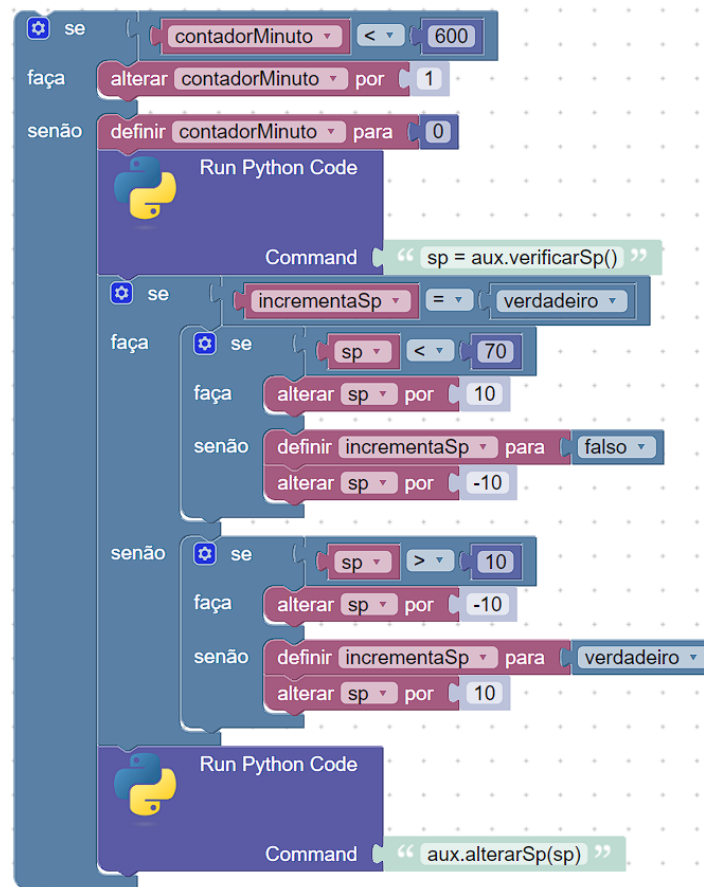


10. Para enviar os dados via serial ou salvar dados no cartão SD, ainda dentro do laço de repetição infinito, insira os blocos a seguir, logo abaixo do último bloco Run Python Code desenvolvido na etapa anterior:

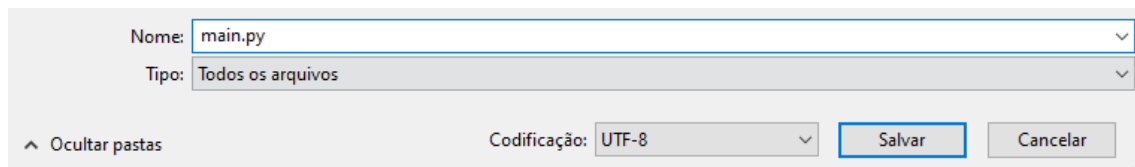


Atenção o comando completo do último bloco é: `aux.salvarDados(aux.formatarDados(tempo500ms, nivel, saida))`

11. (Opcional) Para efeito de teste a cada cinco minutos o setpoint é alterado de dez unidade, variando em um range de 10 a 70 %, caso queira implementar essa funcionalidade, inclua os blocos a seguir dentro do laço infinito, logo após os blocos desenvolvidos na etapa anterior dentro do bloco **se dados**.



12. Para que seu código seja iniciado ao ligar ou reiniciar a placa, é necessário gravar o código gerado pelo Bipes em um arquivo com o nome `main.py`. Abra um editor de texto simples como o bloco de notas, por exemplo, acesse a aba **Arquivos** e clique sobre o arquivo `code.bipes.py`, copie todo código gerado pelo Bipes e cole no editor de texto. Salve o arquivo com o nome `main.py` em algum local do seu computador, não esqueça de selecionar em Tipo: Todos os Arquivos.



Carregue o arquivo na placa, ao final, os arquivos da placa deverão ser conforme a próxima figura.

