

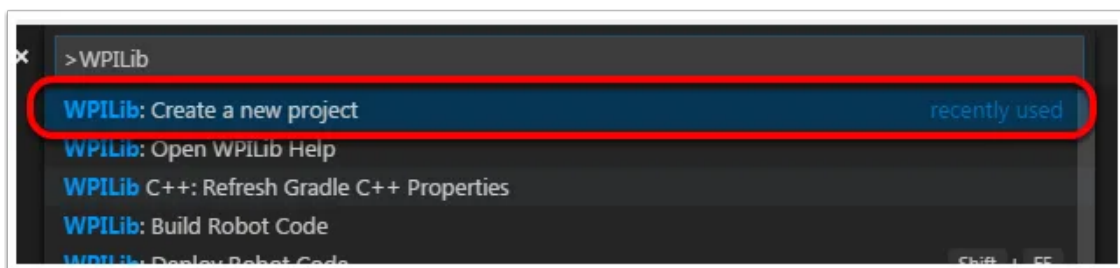
# Criando seu programa de teste (C++/Java)

Assim que tudo estiver instalado, estamos prontos para criar um programa de robô. O WPILib vem com vários modelos para programas de robô. O uso desses modelos é altamente recomendado para novos usuários; no entanto, usuários avançados têm a liberdade de escrever seu próprio código de robô do zero. Este artigo guia você na criação de um projeto a partir de um dos exemplos fornecidos, que já possui algum código escrito para controlar um robô básico.

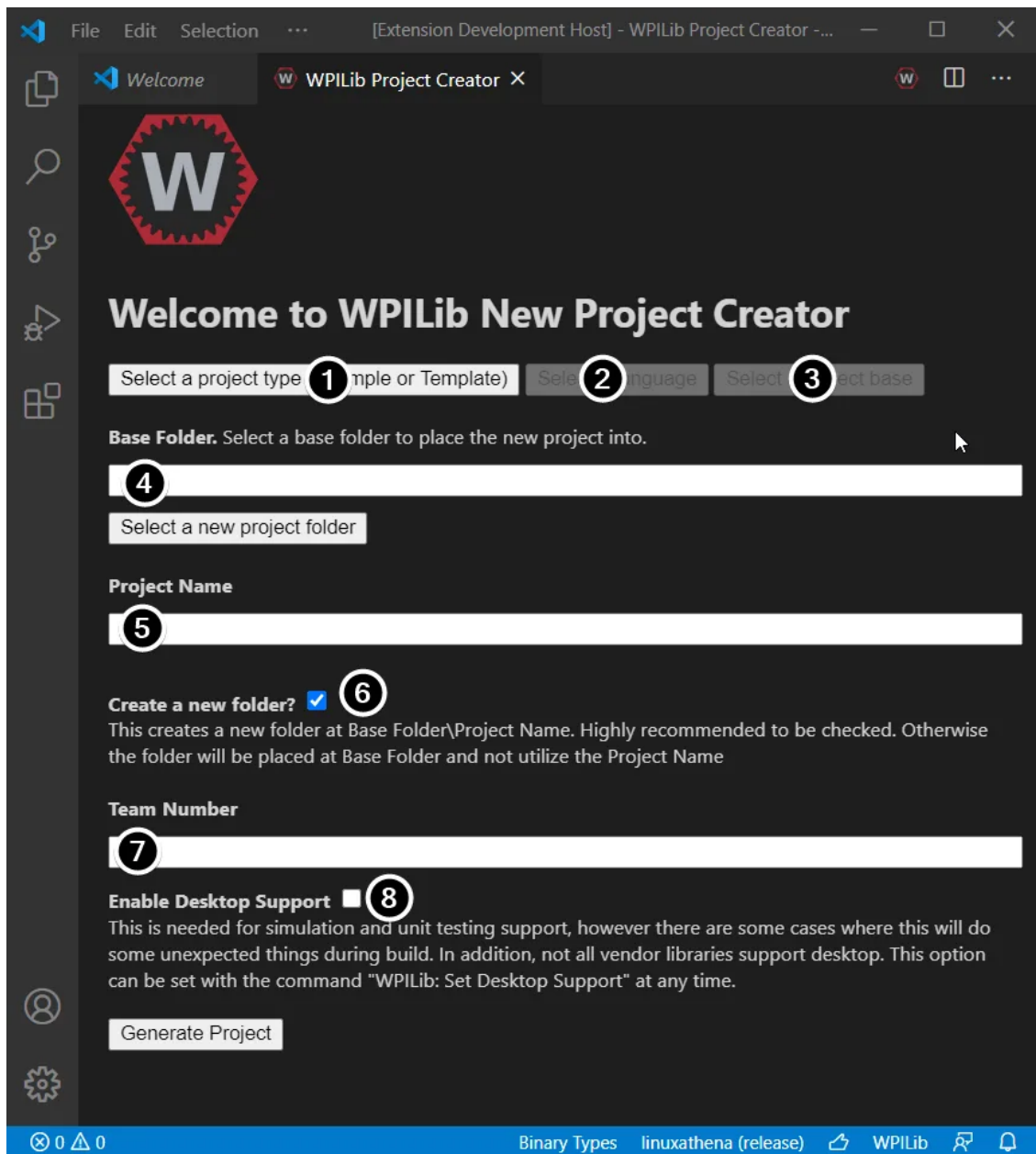
Este guia inclui exemplos de código que envolvem hardware do fornecedor para a conveniência do usuário. Neste documento, PWM refere-se ao controlador de motor incluído no KOP. A guia CTRE faz referência ao controlador de motor Talon FX (motor Falcon 500), mas o uso é semelhante para TalonSRX e VictorSPX. A guia REV faz referência ao CAN SPARK MAX controlando um motor sem escova, mas é semelhante para motor com escovas. Parte-se do pressuposto de que o usuário já instalou os vendedores necessários e configurou o(s) dispositivo(s) (atualização de firmware, atribuição de IDs CAN, etc.) de acordo com a documentação do fabricante (CTRE REV).

## Criando um novo projeto WPILib

Abra a paleta de comandos do Visual Studio Code com Ctrl+Shift+P. Em seguida, digite "WPILib" no prompt. Como todos os comandos do WPILib começam com "WPILib", isso mostrará a lista de comandos específicos do WPILib no VS Code. Agora, selecione o comando "Criar um novo projeto".



Isso abrirá a "Janela do Criador de Novo Projeto:".



Os elementos da "Janela do Criador de Novo Projeto" são explicados abaixo:

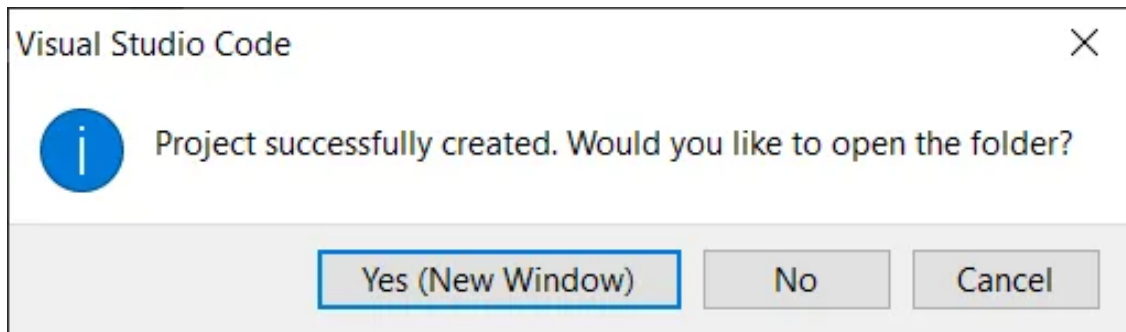
1. Tipo de Projeto: O tipo de projeto que desejamos criar. Para este exemplo, selecione "Exemplo".
2. Linguagem: Esta é a linguagem (C++ ou Java) que será usada para este projeto.
3. Base do Projeto: Esta caixa é usada para selecionar a classe base ou exemplo a ser gerado para o projeto. Para este exemplo, selecione "Início Rápido" (Getting Started).
4. Pasta Base: Isso determina a pasta na qual o projeto do robô será localizado.
5. Nome do Projeto: O nome do projeto do robô. Isso também especifica o nome que será dado à pasta do projeto se a caixa "Criar Nova Pasta" estiver marcada.
6. Criar Nova Pasta: Se esta opção estiver marcada, será criada uma nova pasta para armazenar o projeto dentro da pasta especificada anteriormente. Se não estiver marcada, o projeto será localizado diretamente na pasta especificada anteriormente. Um erro será gerado se a pasta não estiver vazia e esta opção não estiver marcada.
7. Número da Equipe: O número da equipe para o projeto, que será usado para os nomes de pacotes dentro do projeto e para localizar o robô ao implantar o código.

8. Habilitar Suporte de Desktop: Ativa os testes de unidade e a simulação. Embora o WPILib suporte isso, bibliotecas de software de terceiros podem não oferecer suporte a isso. Se as bibliotecas não suportarem desktop, seu código pode não compilar ou pode travar. Deve ser deixado desmarcado, a menos que os testes de unidade ou a simulação sejam necessários e todas as bibliotecas o suportem. Para este exemplo, não marque esta caixa.

Depois de configurar todos os itens acima, clique em "Gerar Projeto" e o projeto do robô será criado.

Quaisquer erros na geração do projeto aparecerão no canto inferior direito da tela.

## Abrindo o novo projeto



Após criar com sucesso o seu projeto, o VS Code dará a opção de abrir o projeto, conforme mostrado acima. Podemos optar por fazer isso agora ou mais tarde, digitando Ctrl+K e, em seguida, Ctrl+O (ou apenas Command+O no macOS) e selecionar a pasta onde salvamos nosso projeto.



## Do you trust the authors of the files in this folder?

Code provides features that may automatically execute files in this folder.

If you don't trust the authors of these files, we recommend to continue in restricted mode as the files may be malicious. See [our docs](#) to learn more.

C:\Users\Joe\Documents\Robotics\RobotBuilderTestProjectCpp-Imported2022Alpha3

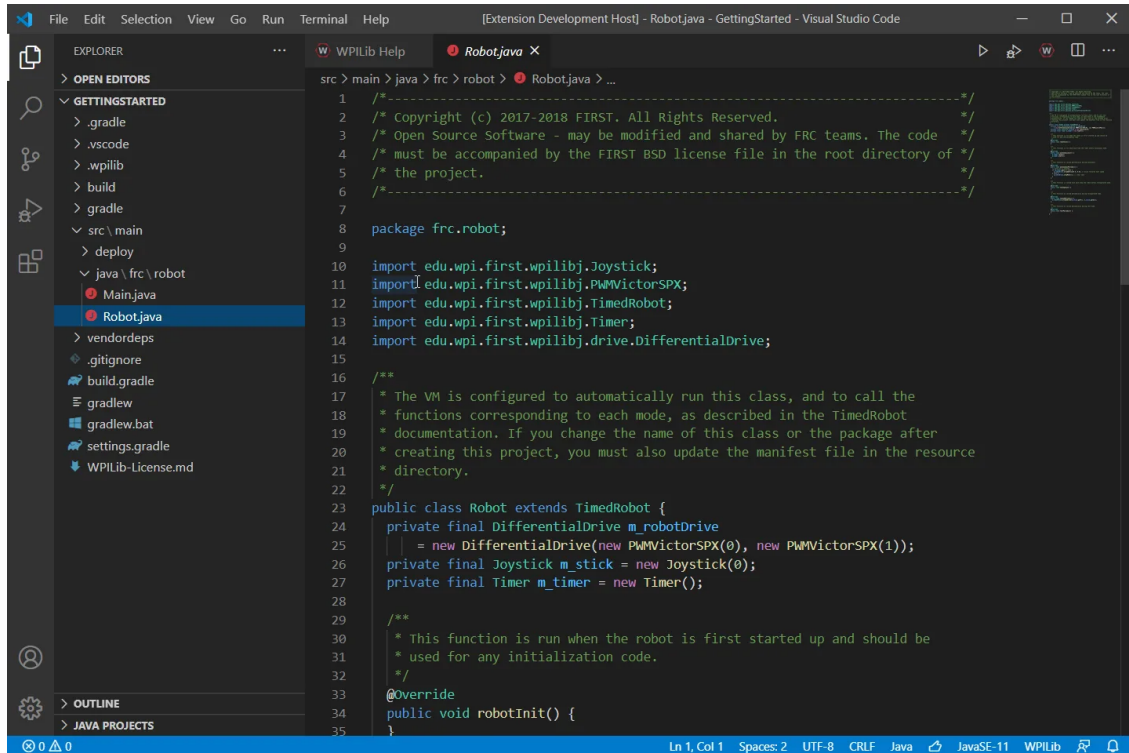
☐ Trust the authors of all files in the parent folder 'Robotics'

**Yes, I trust the authors**  
*Trust folder and enable all features*

**No, I don't trust the authors**  
*Browse folder in restricted mode*

Clique em "Sim, confio nos autores".

Depois de abrir, veremos a hierarquia do projeto à esquerda. Clicar duas vezes no arquivo abrirá esse arquivo no editor.



# Importações

## PWM

```
import edu.wpi.first.wpilibj.TimedRobot;  
import edu.wpi.first.wpilibj.Timer;  
import edu.wpi.first.wpilibj.XboxController;  
import edu.wpi.first.wpilibj.drive.DifferentialDrive;  
import edu.wpi.first.wpilibj.motorcontrol.PWMSparkMax;
```

## CTRE

```
import edu.wpi.first.wpilibj.Joystick;  
import edu.wpi.first.wpilibj.TimedRobot;  
import edu.wpi.first.wpilibj.Timer;  
import edu.wpi.first.wpilibj.drive.DifferentialDrive;  
import com.ctre.phoenix.motorcontrol.can.WPI_TalonFX;
```

## REV

```
import com.revrobotics.CANSparkMax;  
import com.revrobotics.CANSparkMaxLowLevel.MotorType;  
  
import edu.wpi.first.wpilibj.TimedRobot;  
import edu.wpi.first.wpilibj.Timer;  
import edu.wpi.first.wpilibj.XboxController;  
import edu.wpi.first.wpilibj.drive.DifferentialDrive;
```

Nosso código precisa fazer referência aos componentes do WPILib que estão sendo utilizados. Em Java, é feito com declarações import. O programa faz referência a classes como Joystick (para direção), PWMSparkMax / WPI\_TalonFX / CANSparkMax (para controle de motores), TimedRobot (a classe base usada no exemplo), Timer (usado para o modo autônomo) e DifferentialDrive (para conectar o controle do joystick aos motores).

# Definindo as variáveis para nosso robô de exemplo

## PWM

```
public class Robot extends TimedRobot {  
    private final PWMSparkMax m_leftDrive = new PWMSparkMax(0);  
    private final PWMSparkMax m_rightDrive = new PWMSparkMax(1);  
    private final DifferentialDrive m_robotDrive = new DifferentialDrive(m_leftDrive, m_rightDrive);  
    private final XboxController m_controller = new XboxController(0);  
    private final Timer m_timer = new Timer();  
  
    /**  
     * This function is run when the robot is first started up and should be used for any  
     * initialization code.  
     */  
    @Override  
    public void robotInit() {  
        // We need to invert one side of the drivetrain so that positive voltages  
        // result in both sides moving forward. Depending on how your robot's  
        // gearbox is constructed, you might have to invert the left side instead.  
        m_rightDrive.setInverted(true);  
    }  
}
```

## CTRE

```
public class Robot extends TimedRobot {  
    private final WPI_TalonFX m_leftDrive = new WPI_TalonFX(1);  
    private final WPI_TalonFX m_rightDrive = new WPI_TalonFX(2);  
    private final DifferentialDrive m_robotDrive = new DifferentialDrive(m_leftDrive, m_rightDrive);  
    private final Joystick m_stick = new Joystick(0);  
    private final Timer m_timer = new Timer();  
}
```

## REV

```
public class Robot extends TimedRobot {  
    private final CANSparkMax m_leftDrive = new CANSparkMax(1, MotorType.kBrushless);  
    private final CANSparkMax m_rightDrive = new CANSparkMax(2, MotorType.kBrushless);  
    private final DifferentialDrive m_robotDrive = new DifferentialDrive(m_leftDrive, m_rightDrive);  
    private final XboxController m_controller = new XboxController(0);  
    private final Timer m_timer = new Timer();  
}
```

O robô de exemplo em nossos exemplos terá um joystick na porta USB 0 para controle tipo arcade e dois motores nas portas PWM 0 e 1 (exemplos do fornecedor usam CAN com IDs 1 e 2). Aqui, criamos objetos do tipo `DifferentialDrive` (`m_robotDrive`), `Joystick` (`m_stick`) e `Timer` (`m_timer`). Esta seção do código faz três coisas:

1. Define as variáveis como membros da nossa classe `Robot`.
2. Inicializa as variáveis.

## Inicialização do robô

```
@Override  
public void robotInit() {}
```

O método `RobotInit` é executado quando o programa do robô está iniciando, mas após o construtor. O `RobotInit` para o nosso programa de exemplo não faz nada. Se quisermos executar algo aqui, podemos fornecer o código acima para substituir o padrão.

## Autônomo simples de exemplo

```
/** This function is run once each time the robot enters autonomous mode. */  
@Override  
public void autonomousInit() {  
    m_timer.restart();  
}  
  
/** This function is called periodically during autonomous. */  
@Override  
public void autonomousPeriodic() {  
    // Drive for 2 seconds  
    if (m_timer.get() < 2.0) {  
        // Drive forwards half speed, make sure to turn input squaring off
```

```
m_robotDrive.arcadeDrive(0.5, 0.0, false);  
} else {  
    m_robotDrive.stopMotor(); // stop robot  
}  
}
```

O método `AutonomousInit` é executado uma vez cada vez que o robô faz a transição para o modo autônomo a partir de outro modo. Neste programa, reiniciamos o Timer neste método.

`AutonomousPeriodic` é executado uma vez a cada período enquanto o robô está no modo autônomo. Na classe `TimedRobot`, o período é um tempo fixo, que padrão é 20ms. Neste exemplo, o código periódico verifica se o temporizador é inferior a 2 segundos e, se for, avança a meio velocidade usando o método `ArcadeDrive` da classe `DifferentialDrive`. Se mais de 2 segundos tiverem decorrido, o código para o acionamento do robô.

# Controle por analógico para teleoperado

```
/** This function is called once each time the robot enters teleoperated mode. */  
@Override  
public void teleopInit() {}  
  
/** This function is called periodically during teleoperated mode. */  
@Override  
public void teleopPeriodic() {  
    m_robotDrive.arcadeDrive(-m_controller.getLeftY(), -m_controller.getRightX());  
}
```

Assim como no modo Autônomo, o modo Teleop possui funções `TeleopInit` e `TeleopPeriodic`. Neste exemplo, não temos nada para fazer em `TeleopInit`; ele é fornecido apenas para fins ilustrativos. Em `TeleopPeriodic`, o código utiliza o método `ArcadeDrive` para mapear o eixo Y do Joystick para o movimento para frente/trás dos motores de acionamento e o eixo X para o movimento de virar.

## Modo de teste

```
/** This function is called once each time the robot enters test mode. */  
@Override
```



```
public void testInit() {}
```

```
/** This function is called periodically during test mode. */
```

```
@Override
```

```
public void testPeriodic() {}
```

O Modo de Teste é usado para testar a funcionalidade do robô. Semelhante ao TeleopInit, os métodos TestInit e TestPeriodic são fornecidos aqui apenas para fins ilustrativos.

# Compilando o projeto no robô

Por favor, consulte as instruções [aqui](#) para implantar o programa em um robô.

---

Revisão #1

Criado 19 dezembro 2023 18:01:35 por Enzo Coutinho

Atualizado 19 dezembro 2023 18:21:32 por Enzo Coutinho