

# **Engenharia de Software Automotivo - Princípios e Aplicações**

**Evandro Leonardo Silva Teixeira**  
**Faculdade UnB Gama**  
**Universidade de Brasília**

## Sumário

---

<b>1</b>	<b>Configuração do ambiente de desenvolvimento - Notepad++</b>	<b>3</b>
1.1	Preparação do ambiente . . . . .	3
1.2	Automação utilizando macros . . . . .	9
<b>2</b>	<b>Placa de desenvolvimento - ANEB v1.0</b>	<b>12</b>
2.1	Visão geral . . . . .	12
2.2	Definição das interfaces . . . . .	12
2.3	Compilação e gravação de um programa exemplo . . . . .	16
2.4	Acesso remoto a placa de desenvolvimento . . . . .	18
<b>3</b>	<b>Módulo MCP2515_CAN - Notepad++</b>	<b>20</b>
3.1	Instalação da biblioteca MCP_CAN . . . . .	21
3.2	Exemplo de comunicação entre duas ECUs . . . . .	22
3.3	Conexão e compilação dos programas exemplo . . . . .	25
<b>4</b>	<b>Trampoline RTOS</b>	<b>27</b>
4.1	Toolchain necessário para desenvolvimento . . . . .	27
4.1.1	Instalação e configuração AVR Toolchain . . . . .	28
4.2	Instalação do Trampoline para target AVR . . . . .	28
4.3	Compilação de um programa exemplo . . . . .	30
4.4	Gravação de um programa compilado para Arduino Uno . . . . .	33
4.5	Automação utilizando macros . . . . .	34
<b>5</b>	<b>Módulo MCP2515_CAN</b>	<b>37</b>
5.1	Instalação da biblioteca MCP_CAN . . . . .	37
5.2	Exemplo de comunicação entre duas ECUs . . . . .	39
5.3	Conexão e compilação dos programas exemplo . . . . .	47

# 1 CONFIGURAÇÃO DO AMBIENTE DE DESENVOLVIMENTO - NOTEPAD++

Notepad++ é um editor de texto e de código fonte aberto sobre uma licença pública geral (GPL). Ele suporta várias linguagens de programação sendo geralmente instalado e utilizado no sistema operacional Windows. Ele é capaz de suportar o desenvolvimento de várias linguagens de programação como: C, C++, Java, C#, XML, HTML, PHP, JavaScript, dentre outras.

O Notepad++ permite a instalação de plugins para autocomplemento, busca e substituição com integração de expressões regulares, divisão de tela, zoom, além de possuir suporte para macros.

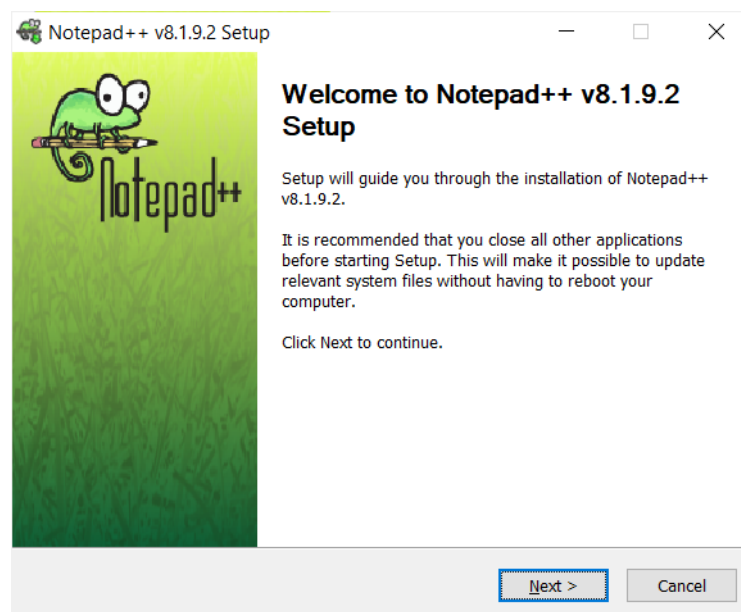
Este tutorial é baseado na instalação das seguintes versões de ferramentas:

- Notepad++ v8.2.1 (32 bits);
- Arduino-cli v0.20.1 (64 bits);

## 1.1 PREPARAÇÃO DO AMBIENTE

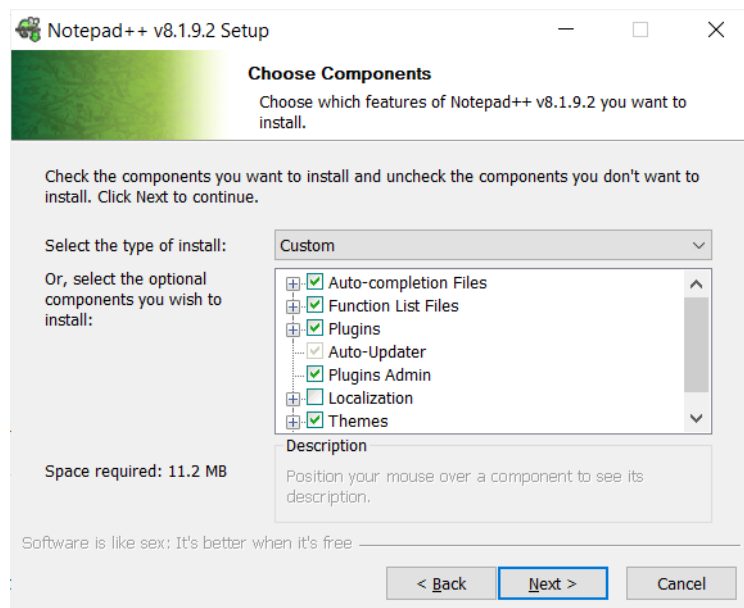
### Passo 1 : Instalar o Notepad++

O Notepad++ a ser instalado é a versão v8.2.1 (32 bits). Isto porque alguns *plugins* a serem utilizados são disponibilizados somente na versão 32 bits. Para instalar o notepad++ basta fazer download através do site: <https://github.com/notepad-plus-plus/notepad-plus-plus/releases/download/v8.2.1/npp.8.2.1.Installer.exe>. Uma vez realizado o download basta clicar no botão próximo (*next*):

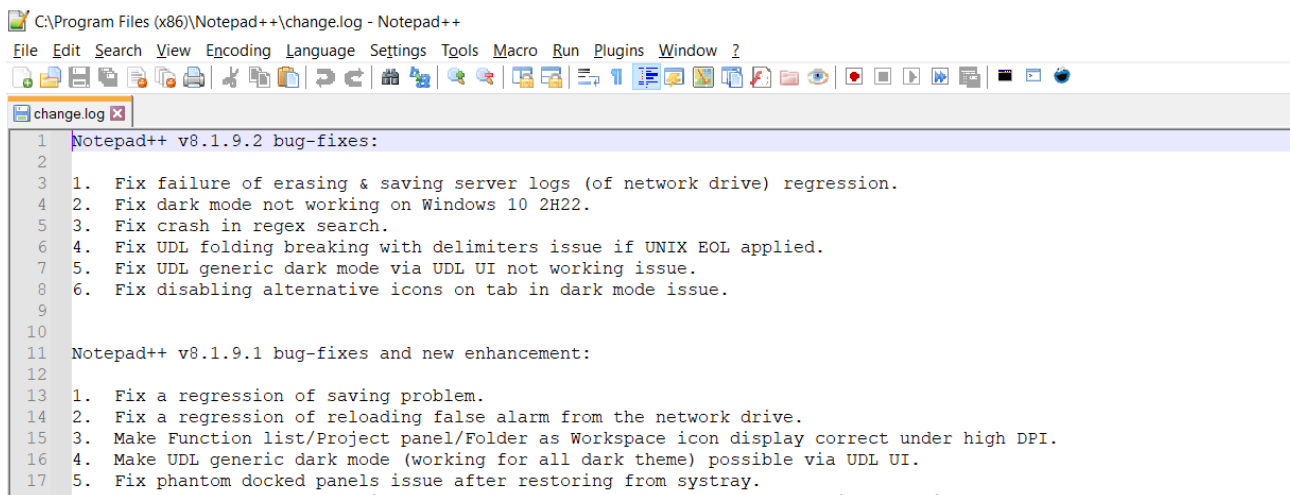


### Passo 2 : Selecionar esta opção custom

Nesta etapa é aconselhável deixar a opção custom selecionada. Isto garante a instalação de grande parte dos recursos necessários para o desenvolvimento.



**Observação importante:** Você deve garantir que todos os itens apresentados (exceto Localization) estejam selecionados. Isto garante uma maior disponibilidade de recursos e uma boa utilização do notepad++. Ao finalizar a instalação e abrir o notepad++ pela primeira vez, você deverá observar a seguinte tela:



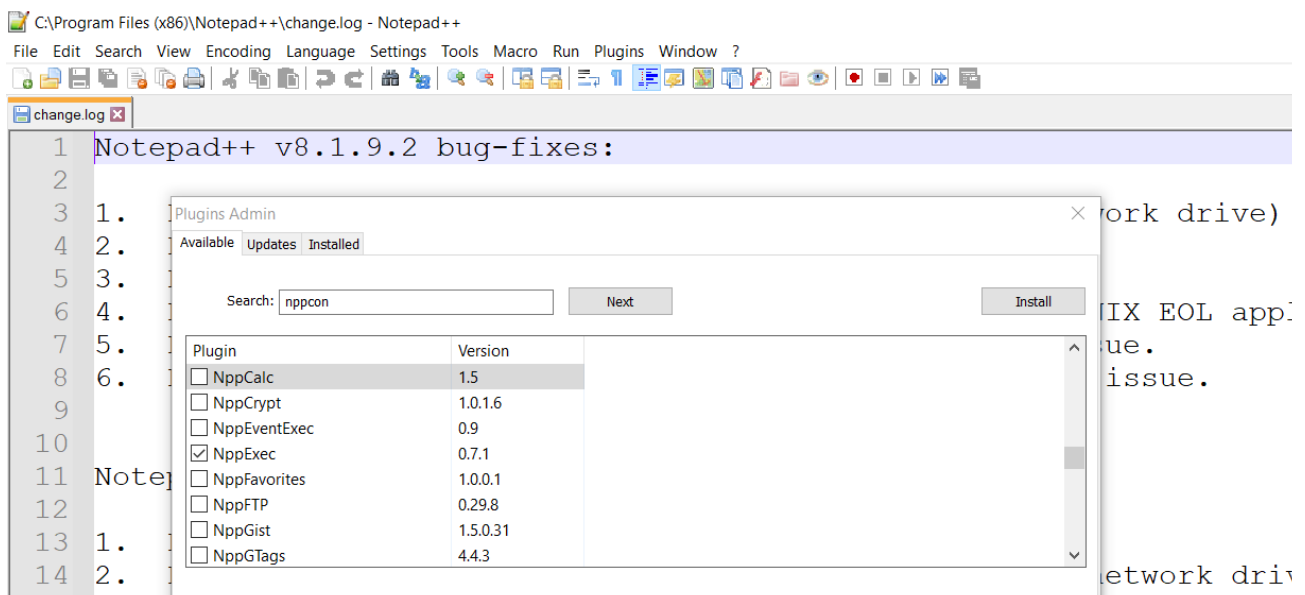
```
C:\Program Files (x86)\Notepad++\change.log - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

change.log
1 Notepad++ v8.1.9.2 bug-fixes:
2
3 1. Fix failure of erasing & saving server logs (of network drive) regression.
4 2. Fix dark mode not working on Windows 10 2H22.
5 3. Fix crash in regex search.
6 4. Fix UDL folding breaking with delimiters issue if UNIX EOL applied.
7 5. Fix UDL generic dark mode via UDL UI not working issue.
8 6. Fix disabling alternative icons on tab in dark mode issue.
9
10
11 Notepad++ v8.1.9.1 bug-fixes and new enhancement:
12
13 1. Fix a regression of saving problem.
14 2. Fix a regression of reloading false alarm from the network drive.
15 3. Make Function list/Project panel/Folder as Workspace icon display correct under high DPI.
16 4. Make UDL generic dark mode (working for all dark theme) possible via UDL UI.
17 5. Fix phantom docked panels issue after restoring from systray.
```

### Passo 3 : Instalar os plugins NppConsole, NppExec e SourceCookfier

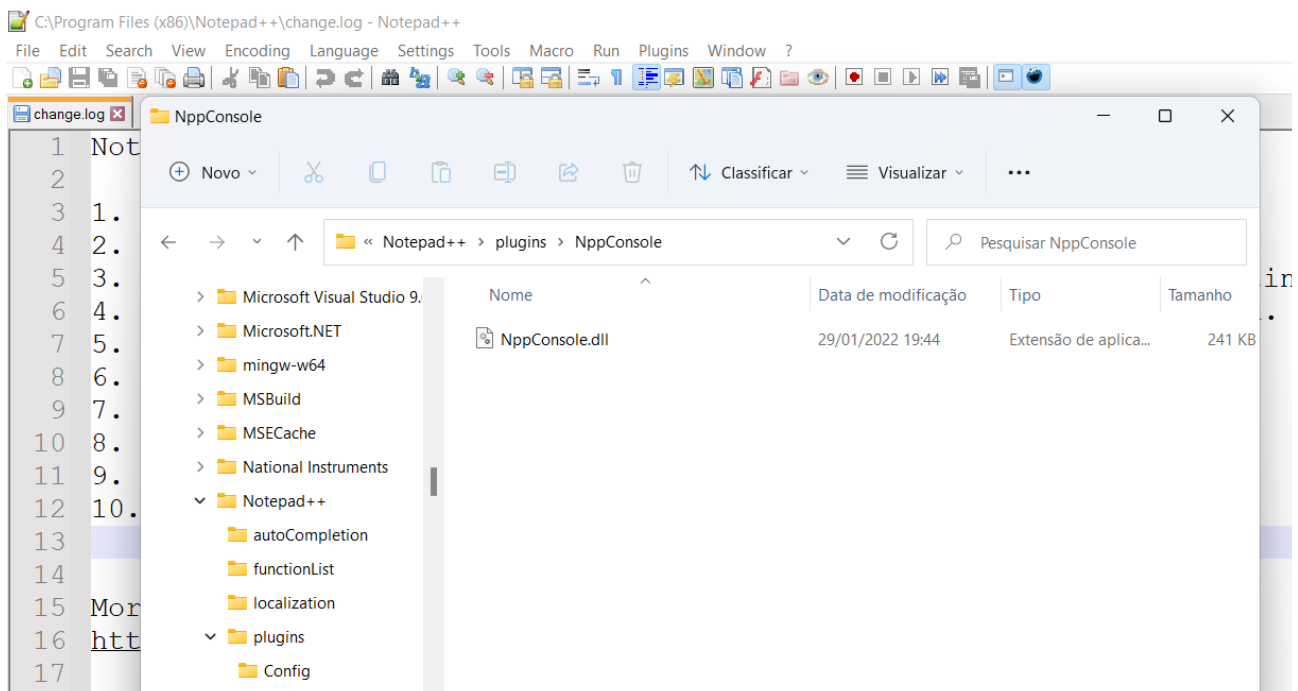
Os plugins a serem instalados permitirão expandir as funcionalidades de uma instalação padrão no Notepad++. Desse modo é fortemente recomendado que você instale os *plugins* NppConsole, NppExec e SourceCookfier para que você consiga utilizar o Notepad++ com todas as funcionalidades necessárias para nos auxiliar no desenvolvimento de software automotivo. Alguns dos plugins sugeridos estão disponíveis em um repositório e poderão ser instalados muito facilmente. Para instalar os plugins NppConsole e NppExec você deve clicar em:

\$ Plugins -> Plugins Admin...



\$ Plugins -> Open Plugins Folder

Posteriormente você deve realizar o download do plugin **NppConsole** acessando o site: <https://sourceforge.net/projects/nppconsole/>, descompactar, e copiar o arquivo **NppConsole.dll** para a pasta NppConsole a ser criada dentro da pasta plugins:



#### Passo 4 : Configurar máscaras de destaque

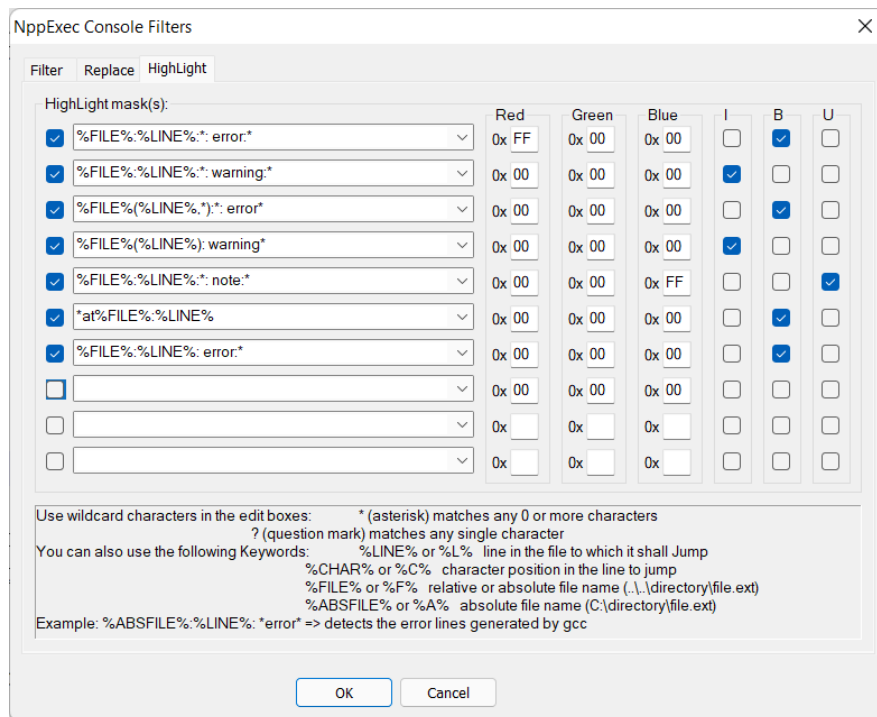
Máscaras de destaque são configuradas para facilitar a identificação e a localização de erros de codificação. Em geral, isto acelera consideravelmente o processo de desenvolvimento de software uma vez que o desenvolvedor conseguirá identificar e localizar rapidamente erros e mensagens de alerta nas linhas código fonte codificadas. Desse modo, para acelerar o desenvolvimento de softwarew automotivo é extremamente recomendado que você realize esta configuração. Para configurar o NppExec Console Filter você deve abrir a janela Console Output Filters clicando em:

\$ Plugins -> NppExec -> Console Output Filters...

Uma vez aberta a janela você deve selecionar a aba *highlights* e inserir um total de sete máscaras de destaque conforme apresentado na tabela abaixo:

N	Máscaras de destaque	Cor	Destaque
1	%FILE%:%LINE%*: error:*	Vermelho (Red = 0xFF)	Negrito (B)
2	%FILE%:%LINE%*: warning:*	-	Itálico (I)
3	%FILE%(%LINE%,)*: error*	-	Negrito (B)
4	%FILE%(%LINE%): warning*	-	Itálico (I)
5	%FILE%:%LINE%*: note:*	Azul (Blue = 0xFF)	Sublinhado (U)
6	*at%FILE%:%LINE%	-	Itálico (I)
7	%FILE%:%LINE%: error:*	-	Itálico (I)

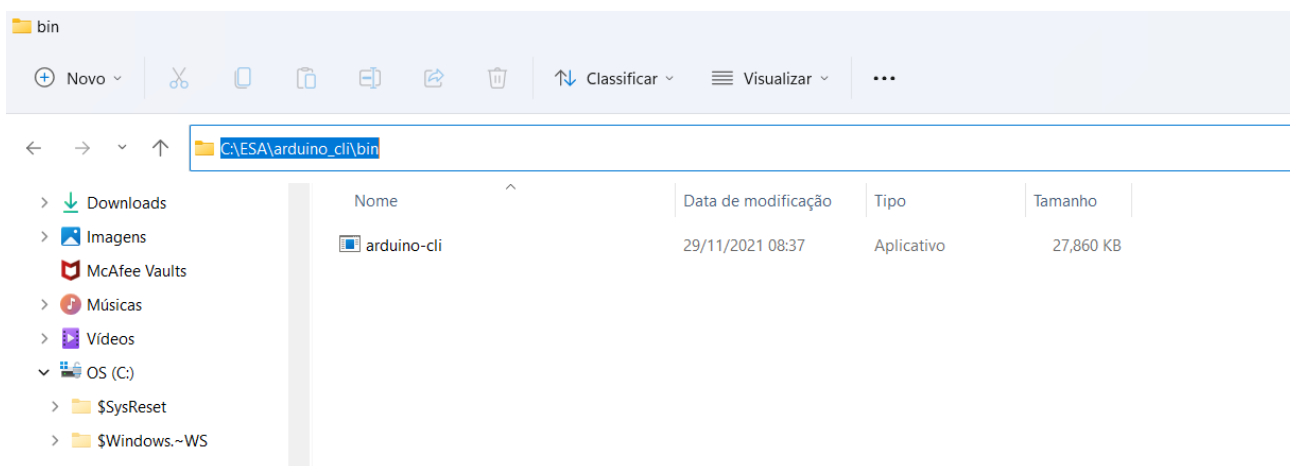
Após realizar corretamente a configuração das máscaras de destaque você deve verificar uma janela conforme destacado conforme figura abaixo. Posteriormente você deve confirmar as alterações realizadas clicando no botão OK:



## Passo 5 : Instalação do Arduino CLI

O Arduino CLI é uma ferramenta de linha de comando utilizada para o desenvolvimento de aplicativos para a plataforma Arduino. Ele serve para analisar a saída JSON da CLI ou implementada como um serviço sempre ativo que aceita comandos por meio de uma interface gRPC. Desse modo, o Arduino CLI facilita a compilação, a geração de imagem e a gravação de firmware para plataforma Arduino. Contudo, nesta etapa iremos configurar o ambiente de desenvolvimento para que ele possa funcionar, de maneira integrada. Para tanto, faça o download do arduino-cli e instale no diretório base conforme ilustrado abaixo:

C:\ESA\arduino\_cli\bin

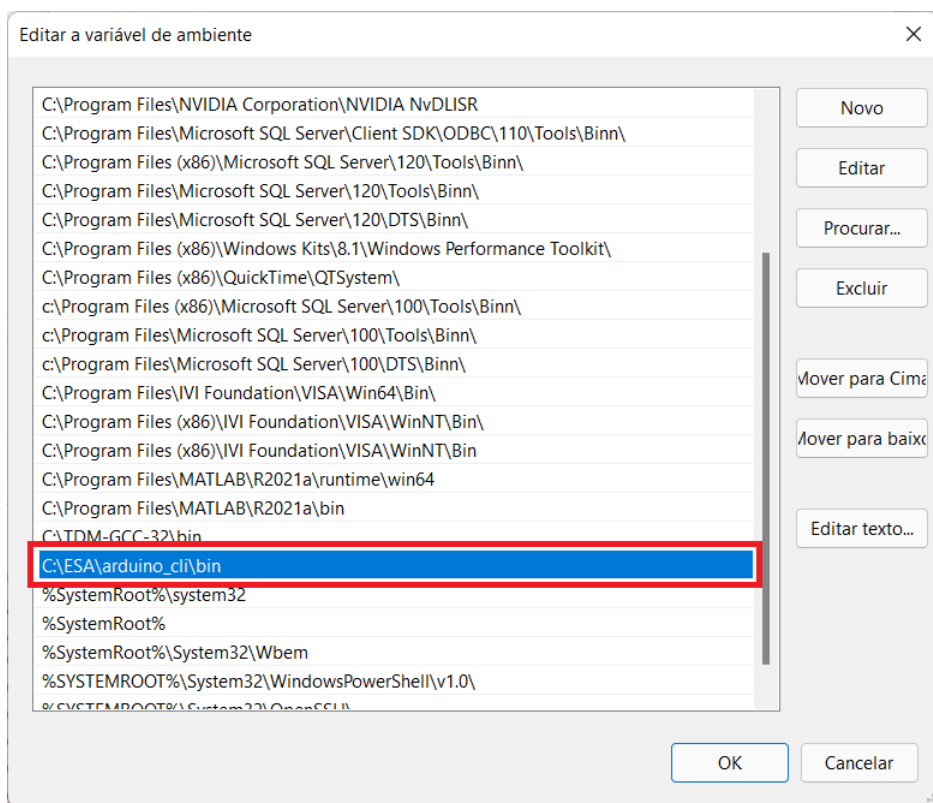


## Passo 6 : Configurando a variável de ambiente PATH

Uma vez adicionado o arquivo **arduino-cli.exe** no diretório base, você deve declarar este diretório na variável de ambiente **PATH**. Esta configuração facilita a execução do comando **arduino-cli**, uma vez que o desenvolvedor não necessita informar o caminho completo para executar o comando. Para adicionar o diretório base do **arduino-cli** na variável de ambiente **PATH**, você deve realizar a seguinte ação:

\$ Digite editar variáveis de ambiente do sistema -> Aba Avançado ->  
Variáveis do Sistema -> Selecione a variável path -> clique em editar

Posteriormente adicione o caminho completo do diretório base na variável **PATH** conforme indicado na figura:



**Observação importante:** Você deve realizar com bastante cuidado a modificação na variável de ambiente **PATH**, uma vez que ela é compartilhada com outros programas instalados em seu computador. Caso tenha alterado, de maneira acidental, algum caminho anteriormente configurado, basta clicar no botão cancelar para suspender a modificação.

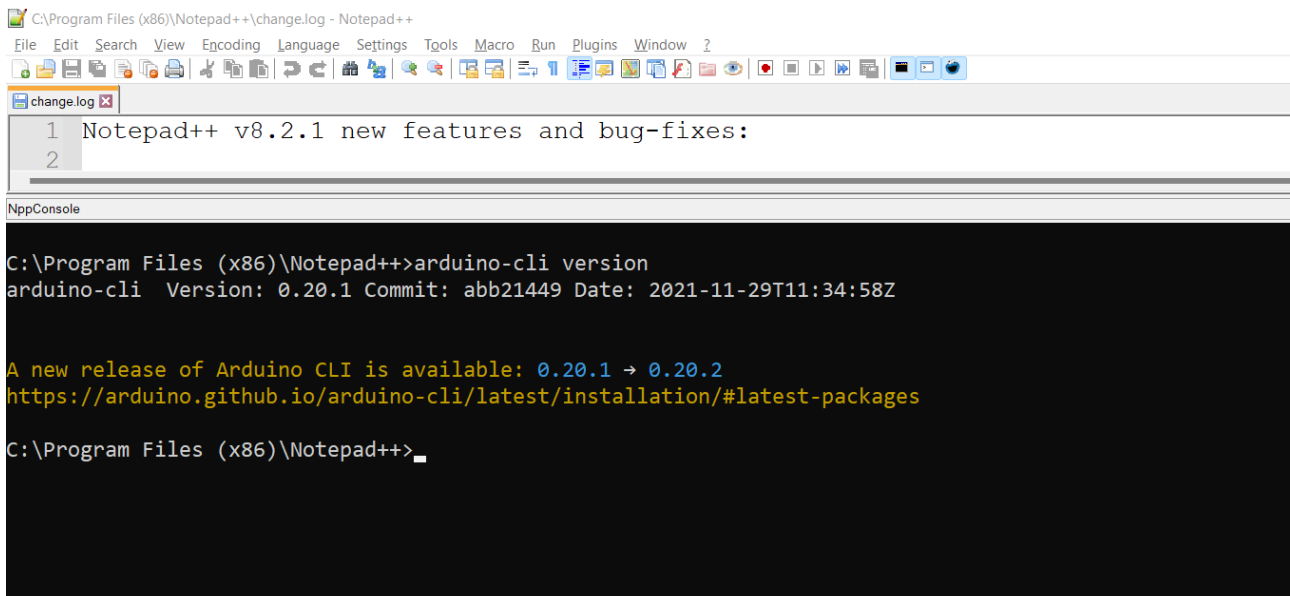
Para testar a configuração da variável de ambiente **PATH**, iremos utilizar o **NppConsole** instalado no **Notepad++**. Para acessar o console no **Notepad++** você deve realizar a seguinte ação:

\$ Plugins -> NppConsole -> Show NppConsole

Posteriormente você deve digitar o seguinte comando no console e verificar o resultado conforme apresentado na figura abaixo:



```
$ arduino-cli version
```



The screenshot shows a Notepad++ window with a file named 'change.log'. The text in the editor is:

```
1 Notepad++ v8.2.1 new features and bug-fixes:
2
```

Below the editor is the NppConsole window, which displays the output of the 'arduino-cli version' command:

```
C:\Program Files (x86)\Notepad++>arduino-cli version
arduino-cli Version: 0.20.1 Commit: abb21449 Date: 2021-11-29T11:34:58Z

A new release of Arduino CLI is available: 0.20.1 → 0.20.2
https://arduino.github.io/arduino-cli/latest/installation/#latest-packages

C:\Program Files (x86)\Notepad++>_
```

## 1.2 AUTOMAÇÃO UTILIZANDO MACROS

As macros são poderosos recursos disponibilizado no Notepad++ pois é a partir delas que podemos automatizar tarefas repetitivas. O Notepad++ permite que o desenvolvedor defina as ações que devem ser automatizadas a partir de macros, salvá-las para uso posterior e então reproduzi-las. Para facilitar o processo de compilação, geração e gravação de imagens no *target* iremos configurar scripts no NppExec.

### Passo 1 : Criando os arquivos scripts

Para iniciar o processo de automação utilizando macros, os scripts `arduino_boards.scp`, `arduino_compile.scp` e `arduino_upload.scp` devem ser criados e inseridos em diretório apropriado. Para tanto, crie três novos arquivos e copie o conteúdo apresentado abaixo em cada um dos arquivos.

#### Arquivo `arduino_boards.scp`

```
//[Arduino] List serial port
cd $(CURRENT_DIRECTORY)
cls
arduino-cli board list
```

#### Arquivo `arduino_compile.scp`

```
//[Arduino] Compile
NPP_SAVEALL
cd $(CURRENT_DIRECTORY)
INPUTBOX "Select the arduino board:":"Board": "$(board)"
set board = $(INPUT)
if "$(board)" == "nano_old" then
```

```

set FQBN = arduino:avr:nano:cpu=atmega328old
set BUILD_DIR = arduino.avr.nano
else if "$(board)" == "nano" then
set FQBN = arduino:avr:nano
set BUILD_DIR = arduino.avr.nano
else if "$(board)" == "uno" then
set FQBN = arduino:avr:uno
set BUILD_DIR = arduino.avr.uno
else if "$(board)" == "m328pb" then
set FQBN = m328pb:avr:atmega328pbcc
set BUILD_DIR = arduino.avr.m328pb
endif
arduino-cli compile -b $(FQBN) -v -e $(FILE_NAME)

```

### Arquivo arduino\_upload.scp

```

//[Arduino] Upload
cd $(CURRENT_DIRECTORY)
INPUTBOX "Select the serial port:":"CommPort": "COM3"
set local commport = $(INPUT)
arduino-cli upload -b $(FQBN) -p $(commport) -v -i build/$(BUILD_DIR)/$(FILE_NAME).hex
-t

```

Posteriormente salve os arquivos criados no passo anteriormente no seguinte diretório:

```
$ C:\Program Files (x86)\Notepad++\plugins\NppExec\scripts
```

### Passo 2 : Configurando as macros para invocar scripts

Nesta etapa você deve configurar a execução dos scripts anteriormente criados. Para tanto você deve abrir a janela execute script clicando em F6 ou executando a seguinte instrução:

```
Plugins -> NppExec -> Execute NppExec Script...
```

Insira as linhas de comando dos scripts abaixo e as salve com os seus respectivos nomes:

#### Linha de comando: script arduino\_boards.scp

```
NPP_EXEC "$(NPP_DIRECTORY)\plugins\NppExec\scripts\arduino_boards.scp"
```

#### Linha de comando: script arduino\_compile.scp

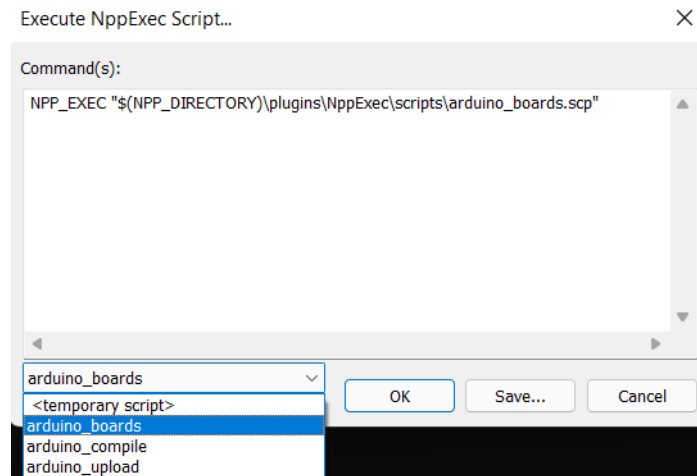
```
NPP_EXEC "$(NPP_DIRECTORY)\plugins\NppExec\scripts\arduino_compile.scp"
```

#### Linha de comando: script arduino\_upload.scp

```
NPP_EXEC "$(NPP_DIRECTORY)\plugins\NppExec\scripts\arduino_upload.scp"
```

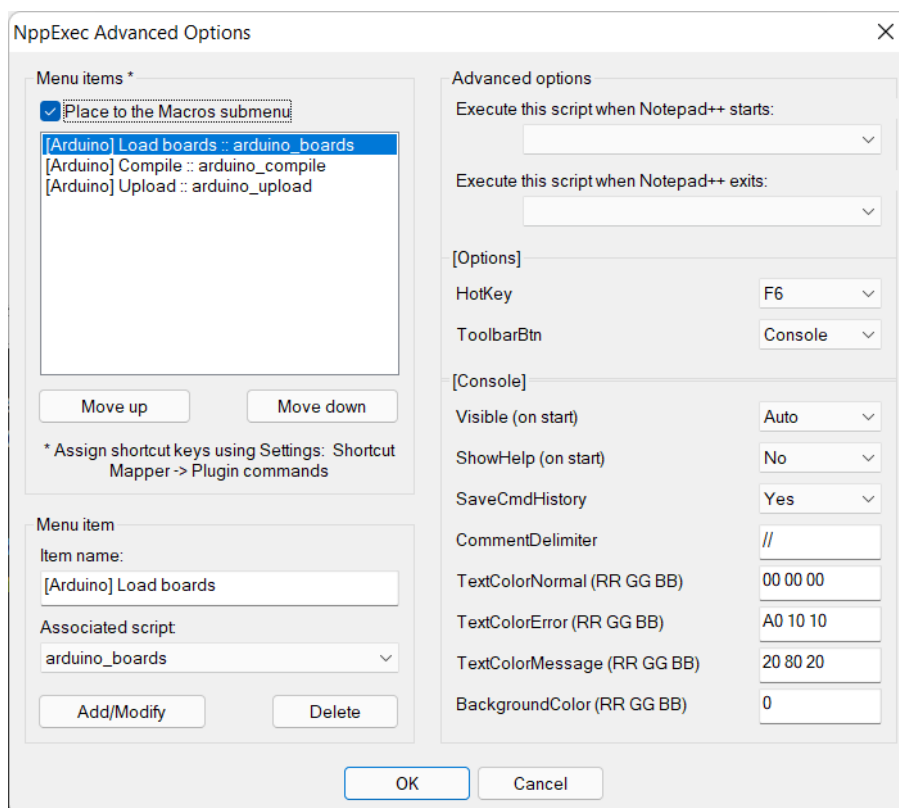
Insira as linhas de comando dos scripts abaixo e as salve com os seus respectivos nomes:

Para configurar a utilização dos scripts por meio de macros vocês devem realizar a seguinte ação:

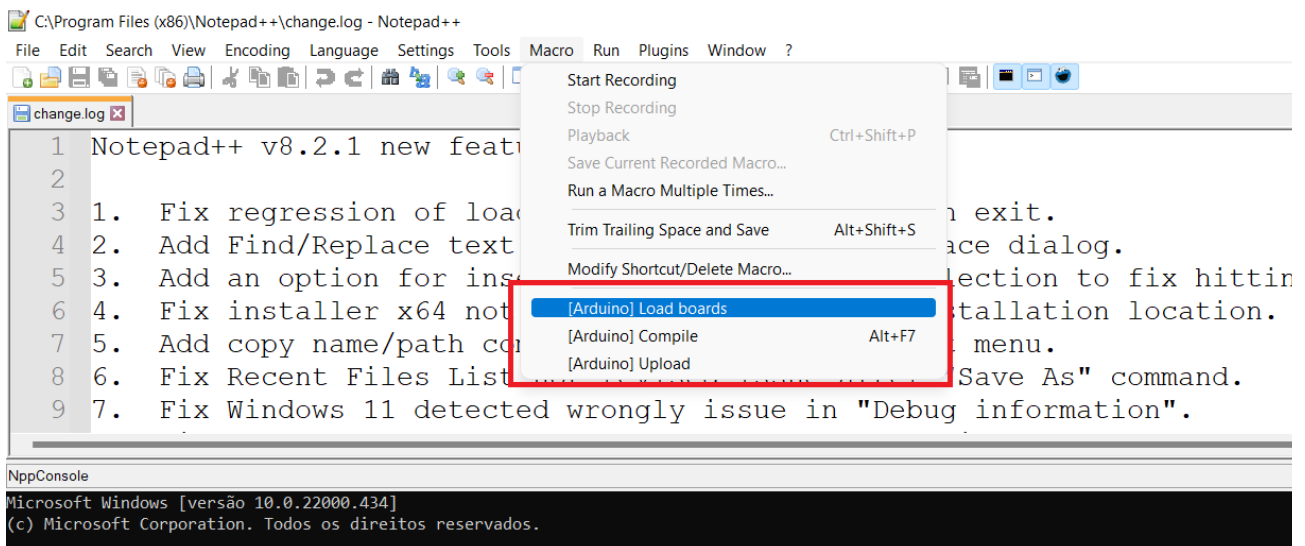


Plugins -> NppExec -> Advanced Options...

Você deve então associar um item de menu a cada um dos scripts associados anteriormente. Para tanto, basta selecionar o script desejado, inserir um nome para o item e clicar no botão Adicionar/Modificar. A figura abaixo apresenta a janela contendo todos os itens de menu associados a cada script.



Contudo para verificar se as macros foram devidamente configuradas acesse o menu Macro para verificar se os itens de menu foram devidamente configurados conforme pode ser observado na figura abaixo:



## 2 PLACA DE DESENVOLVIMENTO - ANEB V1.0

### 2.1 VISÃO GERAL

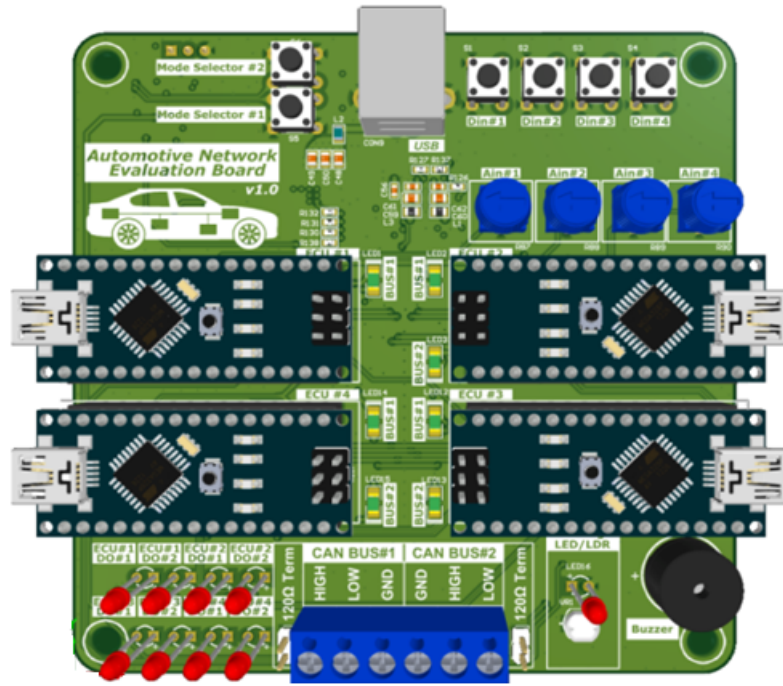
A placa de desenvolvimento é composta pelos seguintes componentes:

- Quatro entradas digitais (pushbuttons);
- Quatro entradas analógicas (potenciômetros - 10 k $\Omega$ );
- Uma saída analógica em loop (ECU1-ECU2);
- Uma saída analógica para LDR (ECU1-ECU1);
- Cinco controladores CAN (8MHz) e transceivers;
- Duas redes CAN (CAN1 e CAN2) independentes;
- Uma subrede automotiva LIN;
- Um hub FTDI para conexão com as ECUs;

### 2.2 DEFINIÇÃO DAS INTERFACES

O arquivo Board.h possui a definição de todos os pinos utilizados na placa de desenvolvimento. Neste sentido, aconselha-se a utilização do referido arquivo no desenvolvimento de aplicações e testes.

#### Arquivo Board.h



```

/////////////////////////////////////////////////////////////////
// Automotive Network Evaluation Board v1.1
// Arquivo: Board.h
//
// Arquivo de definicao da pinagem utilizada pela placa dedesenvolvimento.
// Utilize este arquivo de definicoes para desenvolver aplicacoes
//
/////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////
//1 - ENTRADAS DIGITAIS (PUSHBUTTONS)
/////////////////////////////////////////////////////////////////
//ECU1
#define ECU1_DIN1    MCU_DIN0_PIN
#define ECU1_DIN2    MCU_DIN1_PIN
#define ECU1_DIN3    MCU_DIN2_PIN
#define ECU1_DIN4    MCU_DIN3_PIN
//ECU 2
#define ECU2_DIN1    MCU_DIN0_PIN
#define ECU2_DIN2    MCU_DIN1_PIN
#define ECU2_DIN3    MCU_DIN2_PIN
#define ECU2_DIN4    MCU_DIN3_PIN
//ECU3
#define ECU3_DIN1    MCU_DIN0_PIN
#define ECU3_DIN2    MCU_DIN1_PIN
#define ECU3_DIN3    MCU_DIN2_PIN
#define ECU3_DIN4    MCU_DIN3_PIN
//ECU4
#define ECU4_DIN1    MCU_DIN0_PIN

```

```

#define ECU4_DIN2    MCU_DIN1_PIN
#define ECU4_DIN3    MCU_DIN2_PIN
#define ECU4_DIN4    MCU_DIN3_PIN

////////////////////////////////////
//2 - SAIDAS DIGITAIS
////////////////////////////////////
//ECU1
#define ECU1_DOUT1    MCU_DOUT0_PIN
#define ECU1_DOUT2    MCU_DOUT1_PIN
#define ECU1_BUZZER    MCU1_BUZZ_PIN
//ECU2
#define ECU2_DOUT1    MCU_DOUT0_PIN
#define ECU2_DOUT2    MCU_DOUT1_PIN
//ECU3
#define ECU3_DOUT1    MCU_DOUT0_PIN
#define ECU3_DOUT2    MCU_DOUT1_PIN
//ECU4
#define ECU4_DOUT1    MCU_DOUT0_PIN
#define ECU4_DOUT2    MCU_DOUT1_PIN

////////////////////////////////////
//3 - ENTRADAS ANALOGICAS (POTENCIOMETROS)
////////////////////////////////////
//ECU1
#define ECU1_AIN1    MCU_AIN0_PIN
#define ECU1_AIN2    MCU_AIN1_PIN
#define ECU1_AIN3    MCU_AIN2_PIN
#define ECU1_AIN4    MCU_AIN3_PIN
#define ECU1_LDR_AIN MCU1_AIN0_PIN
//ECU2
#define ECU2_AIN1    MCU_AIN0_PIN
#define ECU2_AIN2    MCU_AIN1_PIN
#define ECU2_AIN3    MCU_AIN2_PIN
#define ECU2_AIN4    MCU_AIN3_PIN
#define ECU2_LOOP_AIN MCU2_AIN0_PIN
//ECU3
#define ECU3_AIN1    MCU_AIN0_PIN
#define ECU3_AIN2    MCU_AIN1_PIN
#define ECU3_AIN3    MCU_AIN2_PIN
#define ECU3_AIN4    MCU_AIN3_PIN
//ECU4
#define ECU4_AIN1    MCU_AIN0_PIN
#define ECU4_AIN2    MCU_AIN1_PIN
#define ECU4_AIN3    MCU_AIN2_PIN
#define ECU4_AIN4    MCU_AIN3_PIN

////////////////////////////////////
//5 - SAIDAS ANALOGICAS (LDR LED E LOOP)
////////////////////////////////////

```

```

#define ECU1_LOOP MCU1_AOUT0_PIN //Varia de 0 a 1023
#define ECU1_LDR_LED MCU1_AOUT1_PIN //Varia 0 a 255

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//6 - REDE CAN (CAN1_BUS E CAN2_BUS)
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//ECU1- Somente CAN1_BUS
#define ECU1_CAN1_CS CAN1_CS_PIN //Chip selector (CS)
#define ECU1_CAN1_INT CAN_INT_PIN //Interrupt pin

//ECU2 - CAN1_BUS e CAN2_BUS
#define ECU2_CAN1_CS CAN1_CS_PIN //Chip selector (CS)
#define ECU2_CAN2_CS CAN2_CS_PIN //Chip selector (CS)
#define ECU2_CAN1_INT CAN_INT_PIN //Interrupt pin
#define ECU2_CAN2_INT CAN_INT_PIN //Interrupt pin

//ECU3 - CAN1_BUS ou CAN2_BUS
#define ECU3_CAN1_CS CAN1_CS_PIN //Chip selector (CS)
#define ECU3_CAN2_CS CAN1_CS_PIN //Chip selector (CS)
#define ECU3_CAN1_INT CAN_INT_PIN //Interrupt pin
#define ECU3_CAN2_INT CAN_INT_PIN //Interrupt pin

//ECU4 - CAN1_BUS ou CAN2_BUS
#define ECU4_CAN1_CS CAN1_CS_PIN //Chip selector (CS)
#define ECU4_CAN2_CS CAN1_CS_PIN //Chip selector (CS)
#define ECU4_CAN1_INT CAN_INT_PIN //Interrupt pin
#define ECU4_CAN2_INT CAN_INT_PIN //Interrupt pin

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//7 - REDE LIN (ECU3 E ECU4)
//Obs: Utilize a biblioteca SoftwareSerial
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//ECU3
#define ECU3_LIN_TX LIN_TX_PIN
#define ECU3_LIN_RX LIN_RX_PIN
//ECU4
#define ECU4_LIN_TX LIN_TX_PIN
#define ECU4_LIN_RX LIN_RX_PIN

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//PIN MAPPING
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#define MCU_DIN0_PIN A4
#define MCU_DIN1_PIN A5
#define MCU_DIN2_PIN 9
#define MCU_DIN3_PIN 8
#define MCU1_BUZZ_PIN 7

#define MCU_DOUT0_PIN 3
#define MCU_DOUT1_PIN 4

```

```

#define MCU_AIN0_PIN A0
#define MCU_AIN1_PIN A1
#define MCU_AIN2_PIN A2
#define MCU_AIN3_PIN A3

#define MCU1_AIN0_PIN A6
#define MCU2_AIN0_PIN A6
#define MCU2_AIN0_PIN A7

#define MCU1_AOUT0_PIN 5
#define MCU1_AOUT1_PIN 6

#define CAN1_CS_PIN 10
#define CAN2_CS_PIN 6
#define CAN_INT_PIN 2

//Lin tx and rx pin
#define LIN_TX_PIN 5
#define LIN_RX_PIN 6

```

## 2.3 COMPILAÇÃO E GRAVAÇÃO DE UM PROGRAMA EXEMPLO

Nesta seção iremos compilar um simples programa exemplo para verificar se o Notepad++ foi devidamente configurado. Para tanto iremos utilizar o programa exemplo blink.ino comumente disponibilizado no Arduino IDE. Basicamente, este código fonte liga e desliga um led periodicamente considerando o período de 1 segundo.

### Arquivo blink.ino

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Automotive Network Evaluation Board v1.0 (Arquivo Blink.h)
// Pisca um led a cada 1 segundo
// Exemplo de domínio público
// https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#include "board.h"

// Executa somente uma vez após pressionar o botão reset ou ligar
void setup() {
    // Inicializa o pino digital como entrada.
    pinMode(LED_BUILTIN, OUTPUT);
}

//Executa continuamente
void loop() {
    //Liga o led
    digitalWrite(LED_BUILTIN, HIGH);
    //Aguarda 500ms

```



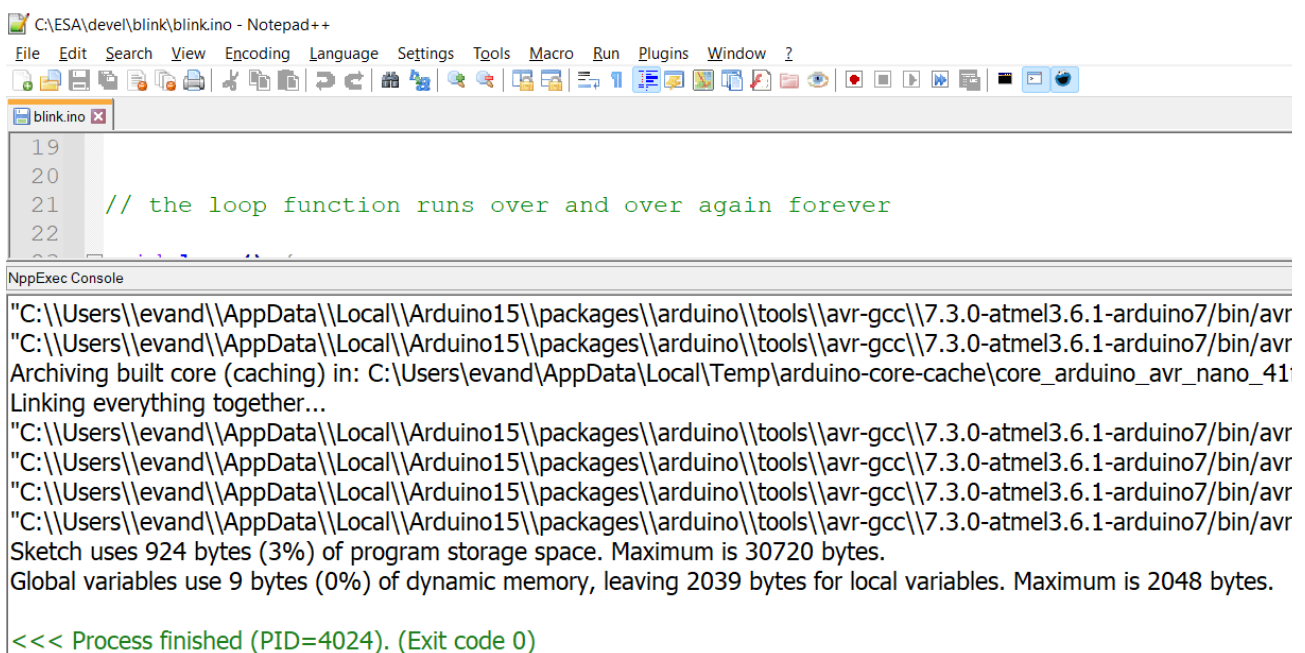
```

delay(500);
//Desliga o led
digitalWrite(LED_BUILTIN, LOW);
//Aguarda 500ms
delay(500);
}

```

Após salvar o arquivo de código-fonte exemplo, selecione o menu compile para que o código-fonte seja compilado. Para verificar se ocorreu sucesso na compilação você deve receber a mensagem conforme descrito na figura abaixo:

Macro -> [Arduino] Compile



```

C:\ESA\devel\blink\blink.ino - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window 2
blink.ino
19
20
21 // the loop function runs over and over again forever
22
NppExec Console
"C:\Users\evand\AppData\Local\Arduino15\packages\arduino\tools\avr-gcc\7.3.0-atmel3.6.1-arduino7/bin/avr
"C:\Users\evand\AppData\Local\Arduino15\packages\arduino\tools\avr-gcc\7.3.0-atmel3.6.1-arduino7/bin/avr
Archiving built core (caching) in: C:\Users\evand\AppData\Local\Temp\arduino-core-cache\core_arduino_avr_nano_41
Linking everything together...
"C:\Users\evand\AppData\Local\Arduino15\packages\arduino\tools\avr-gcc\7.3.0-atmel3.6.1-arduino7/bin/avr
"C:\Users\evand\AppData\Local\Arduino15\packages\arduino\tools\avr-gcc\7.3.0-atmel3.6.1-arduino7/bin/avr
"C:\Users\evand\AppData\Local\Arduino15\packages\arduino\tools\avr-gcc\7.3.0-atmel3.6.1-arduino7/bin/avr
"C:\Users\evand\AppData\Local\Arduino15\packages\arduino\tools\avr-gcc\7.3.0-atmel3.6.1-arduino7/bin/avr
Sketch uses 924 bytes (3%) of program storage space. Maximum is 30720 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.
<<< Process finished (PID=4024). (Exit code 0)

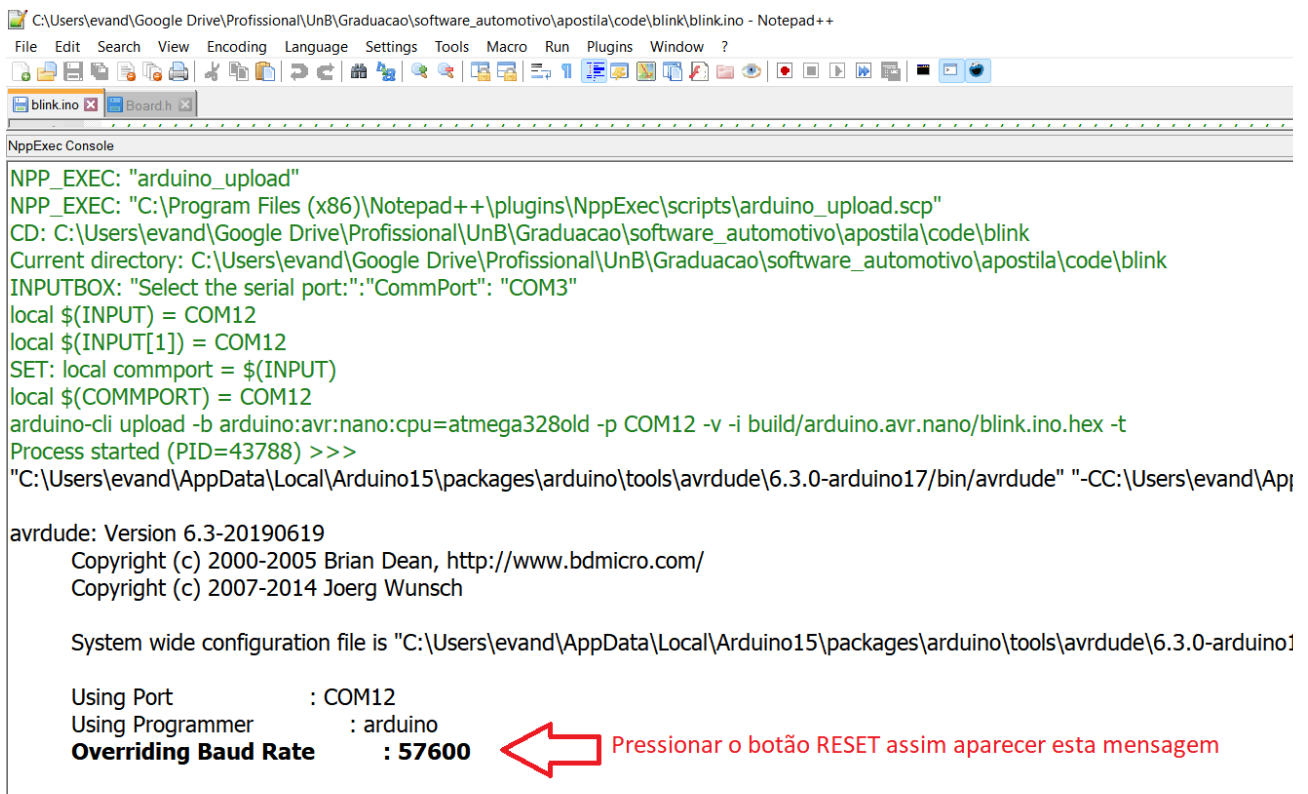
```

**Observação importante:** Para que a compilação ocorra com sucesso você deve inserir o *target* para o qual deseja realizar o upload do programa exemplo.

Após a compilação do arquivo de código-fonte, você deve selecionar a opção upload para gravar o firmware na placa desejada. Para tanto:

Macro -> [Arduino] Upload

Observe que durante a compilação, o script **arduino\_upload.scp** irá solicitar que você insira a porta serial (COM) a qual deseja se conectar para iniciar a gravação. Os nomes atribuídos as portas seriais das ECUs podem variar de computador para computador, bem como de conexão para conexão. Importante salientar que a sequência de portas seriais segue a mesma numeração as ECUs. Por exemplo, ECU1 -> COM13; ECU2 -> COM14; ECU3 -> COM15; ECU4 -> COM16. Não se esqueça de pressionar o botão reset conforme ilustrado na figura abaixo.



```
NPP_EXEC: "arduino_upload"
NPP_EXEC: "C:\Program Files (x86)\Notepad++\plugins\NppExec\scripts\arduino_upload.scp"
CD: C:\Users\evand\Google Drive\Profissional\UnB\Graduacao\software_automotivo\apostila\code\blink
Current directory: C:\Users\evand\Google Drive\Profissional\UnB\Graduacao\software_automotivo\apostila\code\blink
INPUTBOX: "Select the serial port:":"CommPort": "COM3"
local $(INPUT) = COM12
local $(INPUT[1]) = COM12
SET: local commport = $(INPUT)
local $(COMMPORT) = COM12
arduino-cli upload -b arduino:avr:nano:cpu=atmega328old -p COM12 -v -i build/arduino.avr.nano/blink.ino.hex -t
Process started (PID=43788) >>>
"C:\Users\evand\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-arduino17/bin/avrdude" "-CC:\Users\evand\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-arduino17/bin/avrdude.conf"
avrdude: Version 6.3-20190619
Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
Copyright (c) 2007-2014 Joerg Wunsch

System wide configuration file is "C:\Users\evand\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-arduino17/bin/avrdude.conf"

Using Port                : COM12
Using Programmer           : arduino
Overriding Baud Rate       : 57600
```

Pressionar o botão RESET assim aparecer esta mensagem

## 2.4 ACESSO REMOTO A PLACA DE DESENVOLVIMENTO

Para a realização de atividades e laboratórios foi disponibilizado o acesso remoto a placa de desenvolvimento. Assim os estudantes poderão realizar o desenvolvimento de software local e testar utilizando este recurso. Para tanto é necessário que o estudante tenha os seguintes recursos:

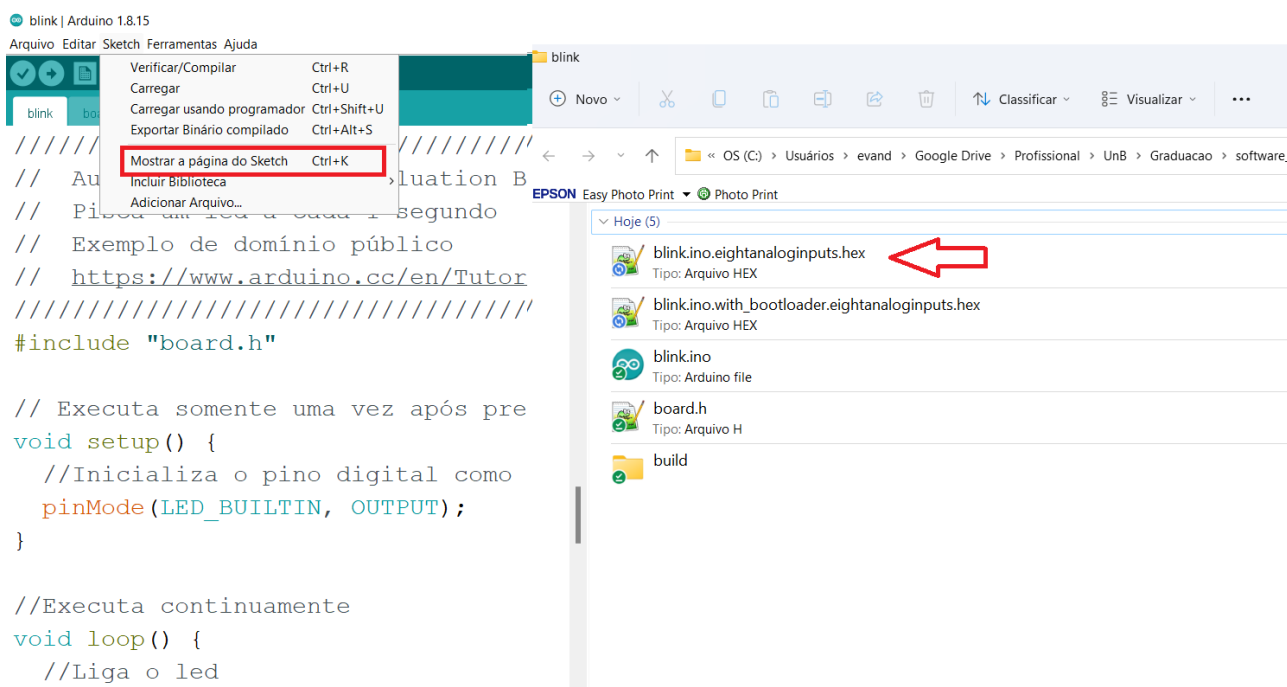
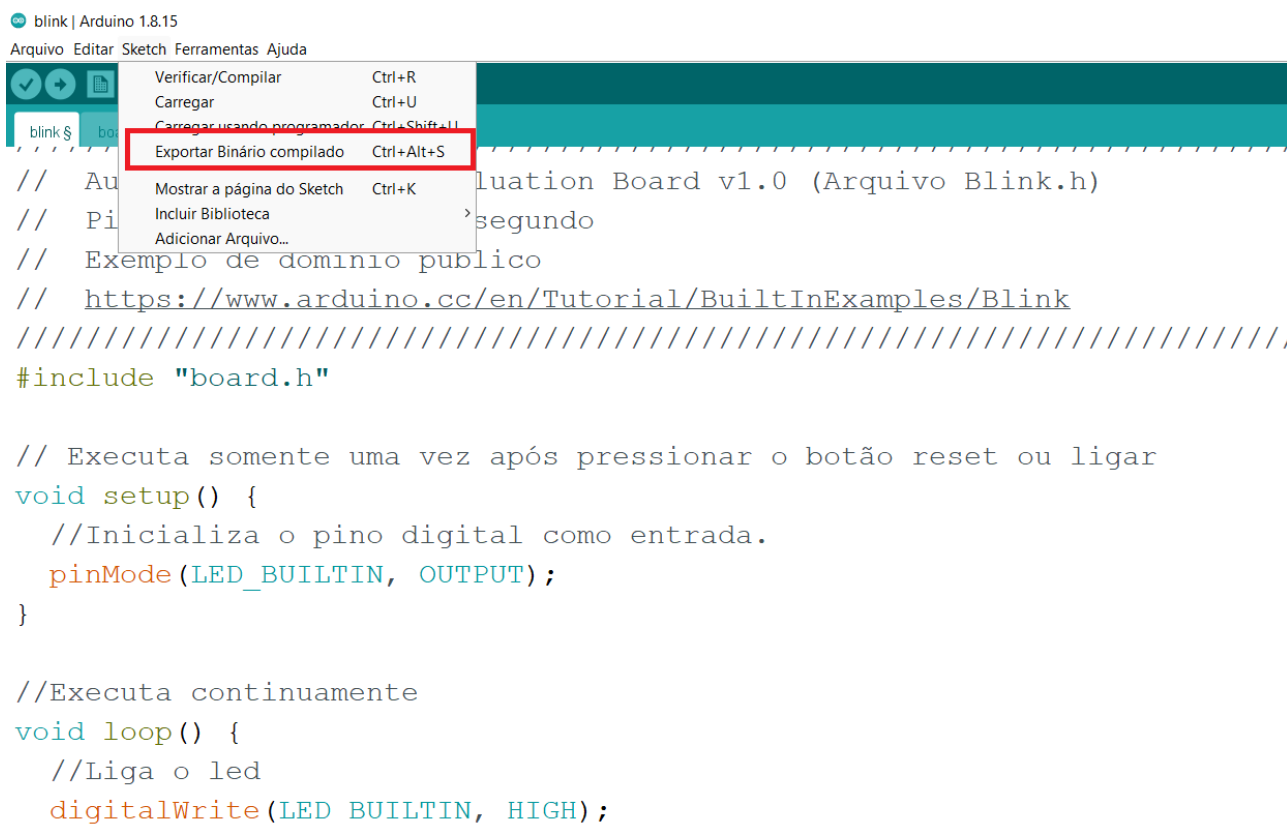
- Computador com o ambiente de desenvolvimento instalado e configurado
- Software Anydesk (versão 7.04 ou inferior)
- Acesso a internet

### Passo 1 : Realizar a compilação do programa

Nesta etapa, o estudante deve realizar a compilação de um programa exemplo. Ele deve gerar o arquivo binário (.hex) do programa. A figura abaixo apresenta este processo na IDE do Arduino:

Posteriormente, você deve abrir a pasta onde o arquivo binário foi exportado. Para tanto, basta acessar o menu Sketch e clicar em mostrar a pasta do Sketch. Este processo é apresentado na figura abaixo:

**Observação importante:** No processo de compilação e geração do arquivo binário é comum que dois arquivos sejam gerados. Você deve gravar o arquivo sem o bootloader (neste caso blink.ino.eightanaloginputs.hex).



## Passo 2 : Conexão com o computador remoto

O arquivo .hex deve ser transferido e gravado na placa conectada ao computador remoto. Para tanto é necessário abrir o programa Anydesk:

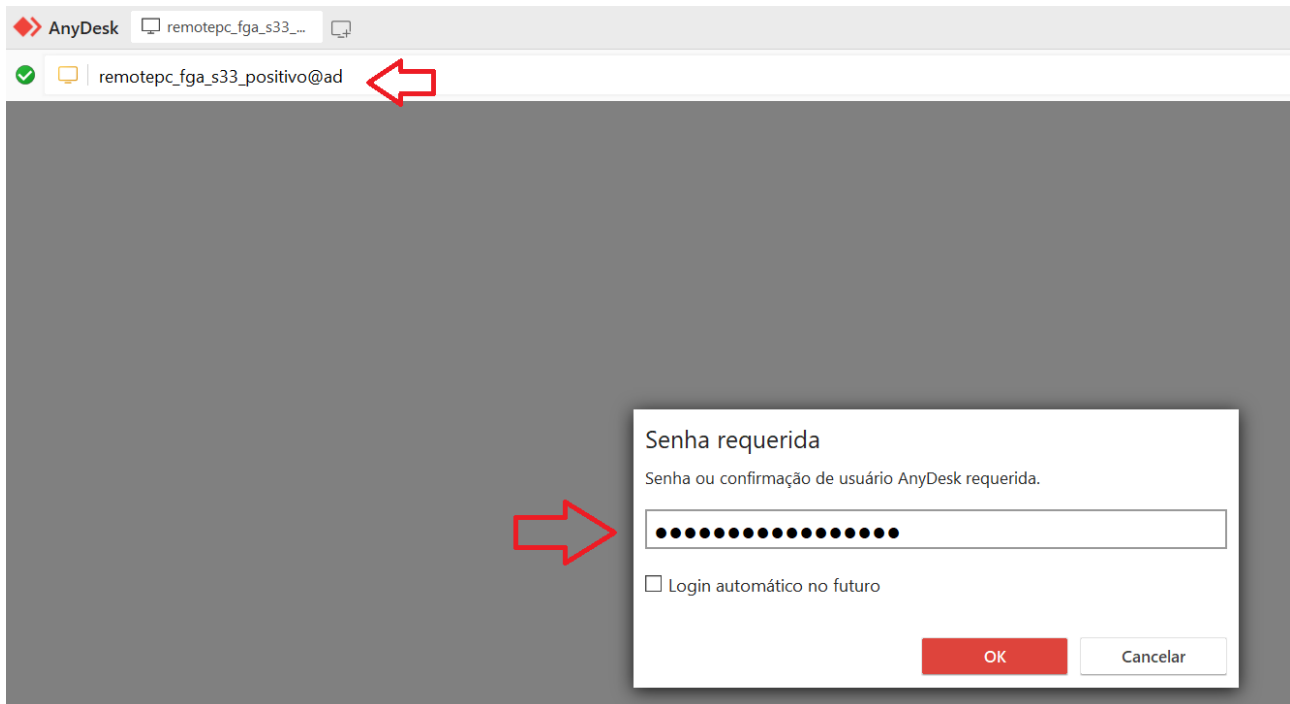
>Iniciar -> Executar -> Anydesk

Posteriormente, você deve estabelecer a conexão com o software Anydesk instalado no

computador remoto. Para tanto, basta realizar a seguinte ação:

- >Catálogo de endereços -> Digite alias do computador remoto
- >Senha requerida -> Digite a senha requerida

**Credenciais 1:** Alias = remotepc\_fga\_s33\_positivo@ad, Senha: remotePCSoft@Auto



### 3 MÓDULO MCP2515\_CAN - NOTEPAD++

O CAN é um protocolo de comunicação serial síncrono de tempo real com alto nível de segurança. Ele é comumente utilizado para integração de ECUs em uma rede de comunicação veicular. O shield MCP2515\_CAN utiliza um controlador CAN MCP2515 que se comunica com um microcontrolado através de uma interface SPI e um transceptor TJA 1050. Utilizando este shield é possível realizar a comunicação entre dois microcontroladores utilizando o protocolo CAN.

Especificações técnicas do shield MCP2515\_CAN:

- Implementa o CAN V2.0B em até 1 Mb/s
- Interface SPI com até 10 MHz
- Frame de dados padrão (11 bits) e estendido (29 bits) e frames remotos
- Dois buffers de recepção com armazenamento prioritário de mensagens

- Tensão de alimentação de 5V
- Resistor de terminação de 120 ohms
- Dois leds indicadores

### 3.1 INSTALAÇÃO DA BIBLIOTECA MCP\_CAN

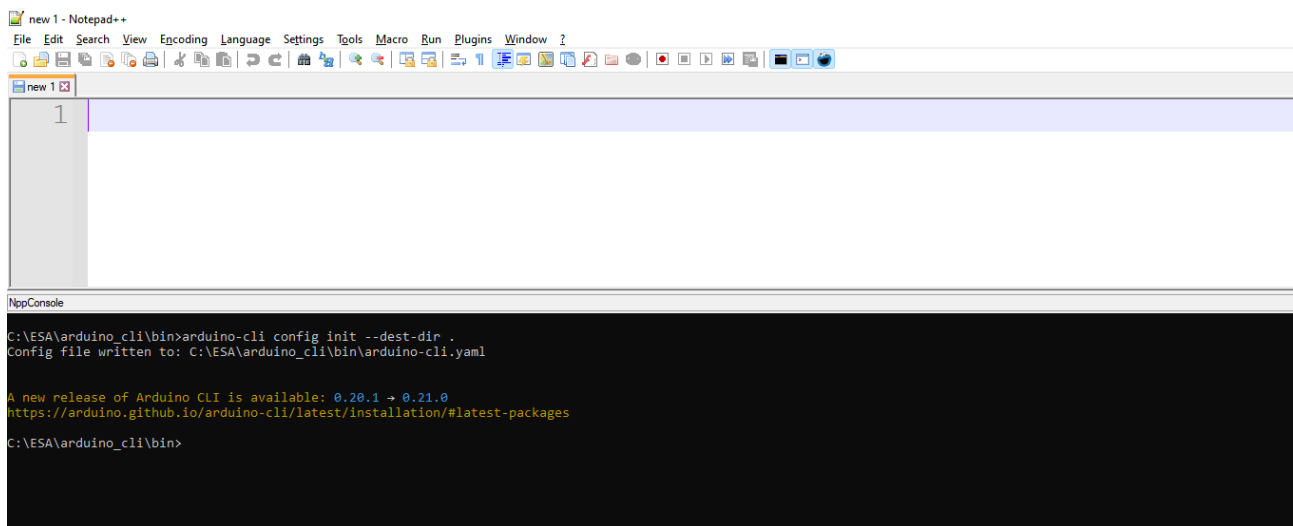
#### Passo 1 : Abra o NppConsole

Clique em Plugins -> NppConsole -> Show NppConsole

#### Passo 2 : Criar o arquivo de configuração do arduino-cli

O arquivo de configuração é utilizado para especificar parâmetros e as configurações iniciais a serem consideradas pelo arduino-cli. Para tanto precisamos modificar este arquivo para instalar a biblioteca libcan. Para tanto, iremos proceder com a criação de um arquivo de configuração personalizado. Para criar o arquivo de configuração você deve mudar para o diretório que contenha o programa arduino-cli e, através do NppConsole, digitar os seguintes comandos.

```
>cd C:\ESA\arduino_cli\bin
>arduino-cli config init --dest-dir .
```



#### Passo 3 : Habilitar a opção instalação não segura no arquivo de configuração

Posteriormente, você deve habilitar a opção instalação não segura. Para tanto, abra o arquivo **arduino-cli.yaml** e modifique a tag **enable\_unsafe\_install** para **true** conforme indicado na imagem abaixo:

#### Passo 4 : Instalação da biblioteca MCP\_CAN

Para utilizar o módulo MCP2515\_CAN é necessário instalar uma biblioteca MCP\_CAN. Existem diversas bibliotecas disponíveis na atualidade. Entretanto, para garantir o correto

```
*C:\ESA\arduino_cli\bin\arduino-cli.yaml - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

arduino-cli.yaml
1 board_manager:
2   additional_urls: []
3 daemon:
4   port: "50051"
5 directories:
6   data: C:\Users\evand\AppData\Local\Arduino15
7   downloads: C:\Users\evand\AppData\Local\Arduino15\staging
8   user: C:\Users\evand\OneDrive\Documents\Arduino
9 library:
10  enable_unsafe_install: true
11 logging:
12   file: ""
13   format: text
14
```

funcionamento iremos utilizar uma biblioteca específica. Desse modo, para instalar a biblioteca MCP\_CAN utilizando o arduino-cli, você deve digitar o seguinte comando:

```
>arduino-cli lib install --git-url https://github.com/coryjfwler/MCP_CAN_lib.git
```

## Passo 5 : Crie os arquivos can\_send e can\_receive.ino

### 3.2 EXEMPLO DE COMUNICAÇÃO ENTRE DUAS ECUS

Neste tutorial utilizaremos dois programas exemplo da própria biblioteca MCP\_CAN para transmitir e receber periodicamente um frame de dados. Basicamente, os programas exemplo emulam duas ECUs conectadas via rede CAN. A ECU1 é responsável por transmitir um frame de dados de forma periódica (500 ms) enquanto que a ECU2 apresenta o frame de dados recebido na interface serial. A figura 4 apresenta esquematicamente o funcionamento dos programas exemplo.

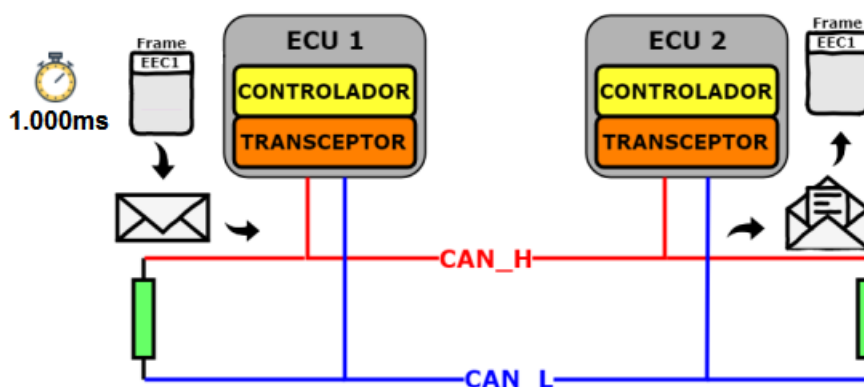


FIGURA 1: ESQUEMA DE FUNCIONAMENTO DO PROGRAMA EXEMPLO

## Passo 6 : Criar e mudar para o diretório de desenvolvimento (devel)

O diretório `devel` é o diretório de desenvolvimento contendo todas as aplicações desenvolvidas neste tutorial. Assim, para criar o diretório execute o seguinte comando:

```
> mkdir C:\ESA\devel\mcp_can_cli
> cd C:\ESA\devel\mcp_can_cli
```

## Passo 7 : Criar os arquivos can\_send.ino e can\_send.ino

O diretório da aplicação consiste em uma pasta criada dentro do diretório de desenvolvimento. Observa-se que no diretório da aplicação são armazenados todos os arquivos necessários para o desenvolvimento, compilação e geração da imagem binária da aplicação. Crie os arquivos `can_send.ino` e `can_receive.ino` preenchê-los com o conteúdo estabelecido as figuras abaixo.

**Arquivo can\_send.ino**

```

// Universidade de Brasilia (UnB)
// Faculdade do Gama
// Curso: Engenharia Automotiva
// Disciplina : Engenharia de Software Automotivo
///////////////////////////////////////////////////////////////////
#include <mcp_can.h>
#include <SPI.h>
#include "Board.h"

//Variavel para armazenar informacoes do frame recebido
byte mData[8]={0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07};

MCP_CAN CAN1(CAN1_CS);

void setup()
{
    // Inicializa a interface serial : baudrate = 115200
    Serial.begin(115200);
    // Inicializa o controlador CAN : baudrate = 500K, clock=8MHz
    //Mascaras e filtros desabilitados .
    while(CAN1.begin(MCP_ANY, CAN_500KBPS, MCP_8MHZ) != CAN_OK)
    {
        delay(500);
        Serial.println("Erro ao inicializar o MCP2515...");
    }
    Serial.println("MCP2515 can_send inicializado com sucesso!");
    //Modifica para o modo normal de operação
    CAN1.setMode(MCP_NORMAL);
}

void loop()
{
    //Envia um frame de dados: ID = 0x100, Std, DLC = 8 bytes,

```

```
//mDATA = array de dados a ser enviado
byte sndStat = CAN1.sendMsgBuf(0x100,CAN_STDID,8,mDATA);
if (sndStat == CAN_OK)
{
    Serial.println("Mensagem enviada com sucesso!");
}
else
{
    Serial.println("Erro para enviar a mensagem...");
}
//Aguarda por 500ms
delay(500);
}
```

## Arquivo can\_receive.ino

```

////////////////////////////////////
// Universidade de Brasilia (UnB)
// Faculdade do Gama
// Curso: Engenharia Automotiva
// Disciplina : Engenharia de Software Automotivo
////////////////////////////////////
#include <mcp_can.h>
#include <SPI.h>
#include "Board.h"

//Variavel para armazenar informacoes do frame recebido
unsigned char mDLC = 0;
unsigned char mData[8];
long unsigned int mID;
char msgString[128];

MCP_CAN CAN1(CAN1_CS);

void setup()
{
    // Inicializa a interface serial : baudrate = 115200
    Serial.begin(115200);

    // Inicializa o controlador CAN : baudrate = 500K, clock=8MHz
    //Mascaras e filtros desabilitados .
    if(CAN1.begin(MCP_ANY,CAN_500KBPS, MCP_8MHZ) != CAN_OK)
    {
        delay(200);
    }
    Serial.println("Modulo CAN inicializado!");
    //Configura o controlador MCP2515 no modo normal.
    CAN1.setMode(MCP_NORMAL);
    //Configura o pino de interrupção para recepção

```



```

pinMode(CAN_INT, INPUT);
Serial.println("MCP2515 exemplo can_receive...");
Serial.print("ID\t\tType\tDLC\tByte0\tByte1\tByte2");
Serial.println("\t\tByte3\tByte4\tByte5\tByte6\tByte7");
}
void loop()
{
    //Se uma interrupção ocorreu interrupção (CAN_INT pino = 0), lê o buffer de recepção
    if (!digitalRead(CAN_INT))
    {
        //Lê os dados: mID = identificador, mDLC = comprimento, mData = dados do
        //frame
        CAN1.readMsgBuf(&mID, &mDLC, mData);

        //Determina se o frame é do tipo standard (11 bits) ou estendido (29 bits)
        if ((mID & CAN_IS_EXTENDED) == CAN_IS_EXTENDED)
        {
            sprintf(msgString, "0x%.8lX\t\tExt\t[%1d]\t", (mID & CAN_EXTENDED_ID),
                mDLC);
        }
        else
        {
            sprintf(msgString, "0x%.3lX\t\tStd\t[%1d]\t", mID, mDLC);
        }

        Serial.print(msgString);
        // Verifica se a mensagem é do tipo de requisição remota.
        if ((mID & CAN_IS_REMOTE_REQUEST) == CAN_IS_REMOTE_REQUEST)
        {
            sprintf(msgString, "rrf");
            Serial.print(msgString);
        }
        else
        {
            for(byte i = 0; i<mDLC; i++)
            {
                sprintf(msgString, "0x%.2X\t", mData[i]);
                Serial.print(msgString);
            }
        }
        Serial.println();
    }
}

```

### 3.3 CONEXÃO E COMPILAÇÃO DOS PROGRAMAS EXEMPLO

Para testar a comunicação entre dois dispositivos via rede CAN é necessário realizar a conexão física entre dois Arduinos e dois módulos MCP2515 conforme especificado na figura

## FIGURA 2: CONEXÃO FÍSICA ENTRE DOIS ARDUINOS VIA REDE CAN



## 4 TRAMPOLINE RTOS

O Trampoline RTOS que é um sistema operacional de tempo real, com escalonamento estático de tarefas, para sistemas embarcados de pequeno porte e com APIs alinhadas com os padrões da indústria automotiva OSEK/VDX OS e AUTOSAR OS 4.2. Ele também possui suporte a diferentes microcontroladores, incluindo arquitetura AVR de 8 bits. Através da implementação de um sistema operacional, o Trampoline oferece diversos recursos de sistemas como tarefas, alarmes, mecanismos de sincronização, dentre outros. Os recursos disponibilizados são especificados por meio de arquivos descritores escritos em linguagem OIL. O Trampoline implementa um escalonador de prioridades estático em que cada tarefa deve possuir uma prioridade única e as tarefas podem ter um escalonamento preemptivo, não preemptivo e preemptivo misto. Trampoline é uma iniciativa acadêmica, de software livre, para implementação do padrão OSEK/VDX RTOS. Foi desenvolvido visando principalmente a portabilidade e a eficiência no uso de memória. Até o momento, o Trampoline suporta a implementação nas seguintes plataformas:

**TABELA 1: ARQUITETURAS COM SUPORTE DO TRAMPOLINE RTOS**

MCU	Architecture	Cores	Evaluation Board
Atmel ATmega328p	8-bit AVR	1	Arduino Uno
Atmel ATmega2560	8-bit AVR	1	Arduino Mega
Atmel SAM D21	Cortex-M0+	1	XPlainedPro
Broadcom BCM2836	Cortex-A7	4	Raspberry Pi 2 Model B
NXP LPC2294	ARM7	1	Olimex LPC-L2294-1MB
NXP / Freescale MK20DX256	Cortex-M4	1	Teensy31
NXP / Freescale MPC564xL	Power Arch	2	XPC56XX EVB + XPC56XL MINI-MODULE
PULPino	RISC-V	1	ZedBoard
STMicroelectronics STM32F4xx	Cortex-M4	1	STM32F4DISCOVERY with STM32F407VG
STMicroelectronics STM32F30x	Cortex-M4	1	Nucleo-32 STM32F303
MicroSemi SmartFusion2	Cortex-M3	1	starterKit

### 4.1 TOOLCHAIN NECESSÁRIO PARA DESENVOLVIMENTO

Este tutorial descreve como instalar o Trampoline RTOS em um computador com um sistema operacional Windows, bem como o passo a passo para desenvolver uma aplicação a ser executada executado em um target AVR (especificamente em placas Arduino UNO e NANO). Observa-se que, em todos os casos, o cross-compilador GCC será utilizado. Observa-se que para executar este tutorial é necessário instalar as seguintes ferramentas para desenvolvimento de aplicações:

**Notepad++:** Notepad++ é um editor de texto e de código fonte aberto sobre uma licença pública geral (GPL). Ele suporta várias linguagens de programação sendo geral-

mente instalado e utilizado no sistema operacional Windows. Pode ser obtido no site: <https://github.com/notepad-plus-plus/notepad-plus-plus/releases/download/v8.2.1/>

**Git for Windows:** consiste em um conjunto nativo e leve de ferramentas que proporciona uma série de recursos do Git SCM para o Windows. Ao mesmo tempo em que fornece interfaces de usuário para usuários experientes e novatos do Git. Pode ser obtido através do site: <https://git-scm.com/download/win>

**Python:** é uma linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, com tipagem dinâmica. Pode ser obtida através do site: <https://www.python.org/downloads/windows/>

**AVR Toolchain:** é uma coleção de ferramentas/bibliotecas utilizadas para desenvolver aplicativos para microcontroladores AVR. Esta coleção inclui bibliotecas, compiladores, montadores, etc. Pode ser obtida através do site: <https://www.microchip.com/mplab/avr-support/avr-and-arm-toolchains-c-compilers>

Este tutorial é baseado na instalação das seguintes versões de ferramentas:

- Notepad++ v8.2.1 (32 bits);
- Git for Windows versão 2.17.1(2);
- Python versão 2.7.14;
- AVR 8-bit Toolchain v3.62 – Windows;

#### 4.1.1 INSTALAÇÃO E CONFIGURAÇÃO AVR TOOLCHAIN

O AVR Toolchain é uma coleção de ferramentas/bibliotecas utilizadas para desenvolver aplicativos para microcontroladores AVR. Esta coleção inclui bibliotecas, compiladores, montadores, etc. Pode ser obtida através do site: <https://www.microchip.com/mplab/avr-support/avr-and-arm-toolchains-c-compilers>. Uma vez feito o download do AVR Toolchain é necessário descompactá-lo em um diretório consistente.

##### Passo 1 : Descompacte o AVR Toolchain

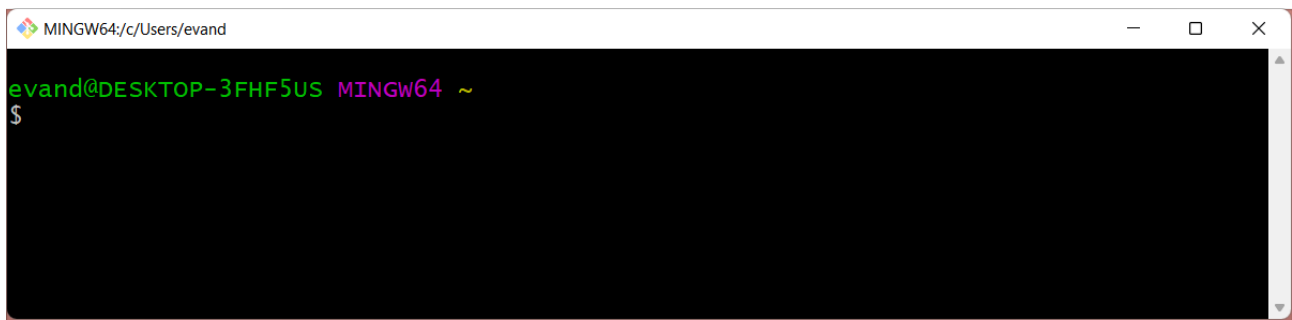
```
> mkdir "C:\ESA\avr_tools"  
> unzip avr8-gnu-toolchain-3.6.2.1778-win32.any.x86.zip -d "C:\ESA\avr_tools"
```

#### 4.2 INSTALAÇÃO DO TRAMPOLINE PARA TARGET AVR

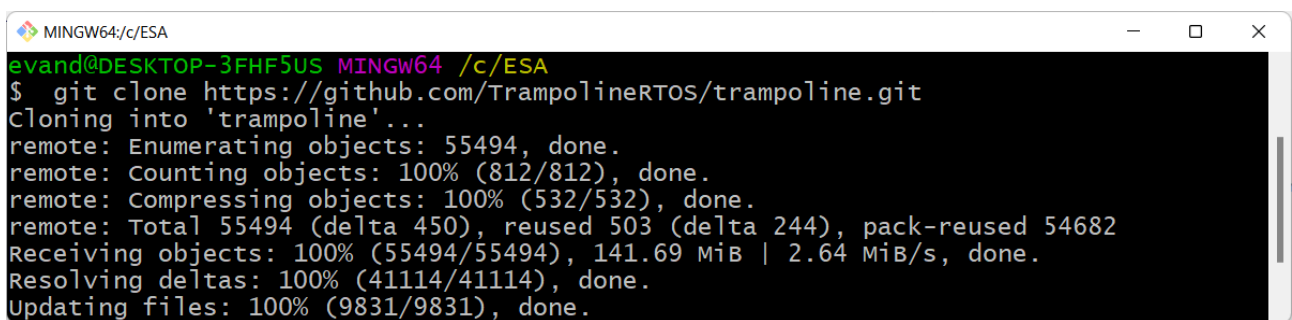
##### Passo 1 : Execute o programa Git bash

Botão iniciar -> buscar programas e arquivos - Git Bash

##### Passo 2 : Mudar para o diretório ESA e realizar download do trampoline

A terminal window titled 'MINGW64:/c/Users/evand' with a black background and green text. The prompt is 'evand@DESKTOP-3FHF5US MINGW64 ~' followed by a '\$' symbol.

```
$ cd "C:\ESA"  
$ git clone https://github.com/TrampolineRTOS/trampoline.git
```

A terminal window titled 'MINGW64:/c/ESA' with a black background and green text. The prompt is 'evand@DESKTOP-3FHF5US MINGW64 /c/ESA'. The output of the 'git clone' command is shown, including progress bars for enumerating, counting, and compressing objects, and resolving deltas.

### Passo 3 : Mudar para o diretório de trabalho

```
$ cd trampoline
```

### Passo 4 : Instalar as bibliotecas do Arduino

Primeiramente é necessário inicializar um submódulo git e posteriormente realizar a atualização do submódulo. Para tanto, digite os seguintes comandos no Git Bash:

```
$ git submodule init  
$ git submodule update machines/avr/arduino
```

Estes comandos permitem obter a versão patched do ArduinoCore-avr para o trampoline localizada no GitHub.

### Passo 5 : Criar o diretório opcional no diretório base do trampoline

```
$ mkdir opt
```

### Passo 6 : Mudar para o diretório opcional

```
$ cd opt
```

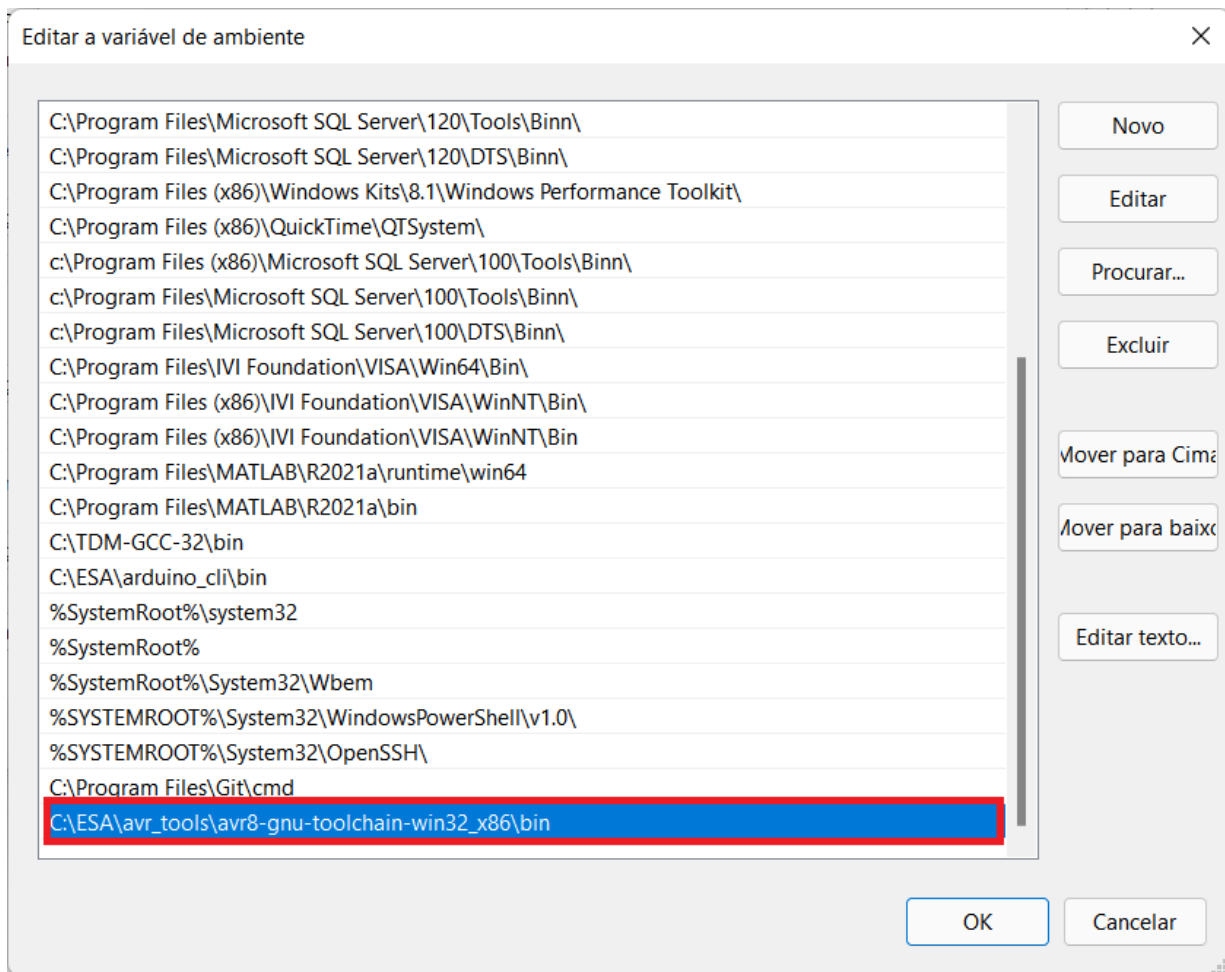
### Passo 7 : Download e descompressão da ferramenta goil for windows:

Realize o download da ferramenta goil for windows e posteriormente realize a descompressão do arquivo no diretório bin do avr\_tools. Goil é o compilador OIL do Trampoline.

```
$ unzip goil-windows.zip -d "C:\ESA\avr_tools\avr8-gnu-toolchain-win32_x86\bin"
```

### Passo 8 : Adicione o `avr_tools` na variável ambiente `PATH`:

Realize procedimento semelhante ao apresentado na seção 1.1.



## 4.3 COMPILAÇÃO DE UM PROGRAMA EXEMPLO

Neste tutorial será utilizado um programa exemplo disponibilizado no diretório *examples* do Trampoline. Este diretório contém diversos exemplos de aplicações para Arduino Uno (MCU ATmega328p) e para o Arduino Mega (MCU ATmega2560). Como o Arduino Nano possui a mesma MCU do Arduino Uno, podemos utilizar o trampoline para desenvolvimento realizando somente algumas alterações. A Lista abaixo apresenta exemplos disponíveis no trampoline:

- **blink:** Pisca um LED utilizando um alarme e uma tarefa.
- **serial:** Exemplo 'blink': utilizando a API serial.
- **extInterrupt:** improve 'serial': add 2 ISRs to change the alarm period.

- **Trace:** The trace toolkit allows to get back kernel events for analysis.

### Passo 1 : Criar um diretório exemplo e mudar para o diretório

Como o Arduino Nano possui a mesma MCU do Arduino Uno faremos uma cópia de todo o código-fonte. Posteriormente acessaremos o diretório criado executando os seguintes comandos:

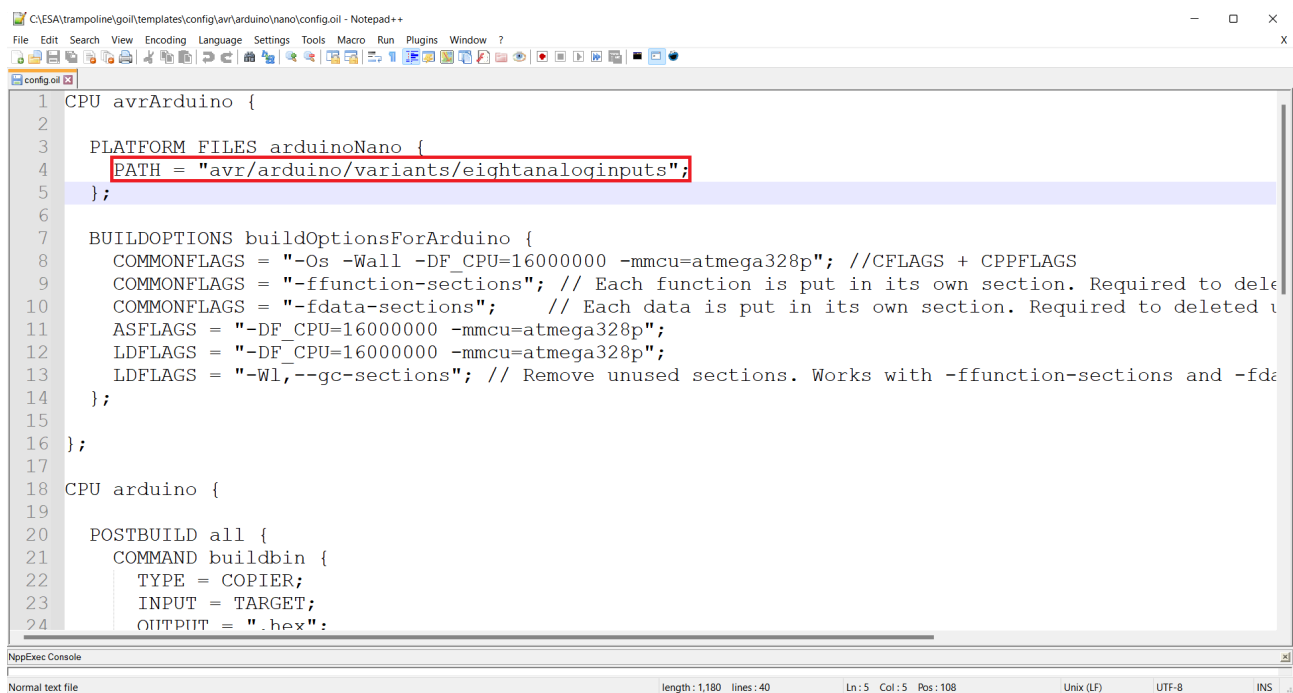
```
$ cp -r ../examples/avr/arduinoUno/ ../examples/avr/arduinoNano
$ cd ../examples/avr/arduinoNano/blink/
```

### Passo 2 : Gerar um template para o Arduino Nano

De maneira semelhante ao passo anterior é necessário criar um template para o Arduino Nano. Para tanto, você deve executar o seguinte comando

```
$ cp -r ../../../../goil/templates/config/avr/arduino/uno/ ../../../../goil/templates/config/avr/arduino/nano
$ cd ../../../../goil/templates/config/avr/arduino/nano
```

Posteriormente, você deve abrir o arquivo config.oil localizado no diretório TRAMPOLINE\_BASE\_PATH/goil/templates/config/avr/nano e realizar a seguinte alteração:



### Passo 3 : Gerar os scripts make.py e build.py

Este passo necessita ser executado somente uma vez. Quando os scripts make.py e build.py foram gerados não é mais necessário executá-lo

```
$ cd goil --target=avr/arduino/nano --templates=../../../../../goil/templates/ blink.oil $ goil
--target=avr/arduino/nano --templates=../../../../../goil/templates/ blink.oil
```

Entendendo os argumentos:


**goil:** compilador OIL (OSEK Implementation Language) do Trampoline

**target:** plataforma alvo (neste caso Arduino uno);

**template:** diretório relativo contendo os templates utilizados na construção dos scripts;

**blink.oil:** Arquivo .oil utilizado para a descrição do RTOS.

**Observação importante:** Caso não haja nenhum erro a mensagem **"No warning, no error."** será apresentada no final do processo de geração.

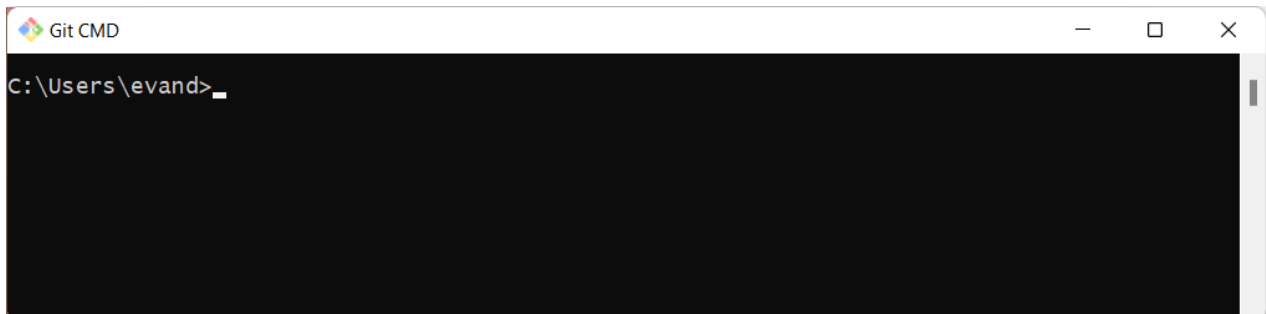


```
MINGW64:/c/esa/trampoline/examples/avr/arduinoNano/blink
Replaced 'blink/tp1_app_config.c'.
Replaced 'blink/tp1_app_config.h'.
Replaced 'blink/tp1_app_define.h'.
Replaced 'blink/tp1_static_info.json'.
Replaced 'blink/tp1_interrupts.c'.
Replaced 'blink/stm_structure.c'.
executing plugin gdb_commands.goilTemplate
Replaced '/c/esa/trampoline/examples/avr/arduinoNano/blink/build/blink.oil.dep'.
No warning, no error.

evand@DESKTOP-3FHF5US MINGW64 /c/esa/trampoline/examples/avr/arduinoNano/blink (master)
$
```

#### Passo 4 : Abrir o prompt de comando

Botão iniciar -> buscar programas e arquivos - Git CMD



```
Git CMD
C:\Users\evand>
```

#### Passo 5 : Mudar para o diretório blink

> cd "C:\ESA\trampoline\examples\avr\arduinoNano\blink"

#### Passo 6 : Executar o script make.py

> make.py

Caso a mensagem: **Generating binary trampuinoBlink.hex from trampuinoBlink** apareça no final da compilação, isto significa que a compilação foi realizada com sucesso tendo como resultado a geração da imagem binária. Observe que a imagem binária consiste na descrição do programa exemplo em linguagem de máquina podendo, desse modo, ser gravada diretamente no *target* (ver seção 4.4).



```
Git CMD
77%] +[1m-[94mCompiling ../../../../machines/avr/arduino/cores/arduino/wMath.cpp-[0m
81%] +[1m-[94mCompiling ../../../../machines/avr/arduino/cores/arduino/wString.cpp-[0m
85%] +[1m-[94mCompiling ../../../../machines/avr/arduino/cores/arduino/wInterrupts.c-[0m
88%] +[1m-[94mCompiling ../../../../machines/avr/arduino/cores/arduino/abi.cpp-[0m
92%] +[1m-[94mCompiling ../../../../machines/avr/arduino/cores/arduino/new.cpp-[0m
96%] +[1m-[94mCompiling ../../../../machines/avr/arduino//tpl_trace.cpp-[0m
100%] +[1m-[94mLinking trampuinoBlink-[0m
+[1m-[94m      Generating binary trampuinoBlink.hex from trampuinoBlink-[0m
C:\ESA\trampoline\examples\avr\arduinoNano\blink>
```

Caso a mensagem de erro: **cc1.exe: error: unrecognized command line option -Wno-unused-but-set-variable**” apareça impedindo a continuidade do processo de compilação é necessário alterar as configurações do buildOptionsForArduino.

### Passo 7 : Alterando as configurações do buildOptionsForArduino

Para tanto abra o arquivo config.oil localizado no seguinte diretório:

```
C:\Users\evandroleonardo\trampoline\goil\templates\config\avr\arduino
```

Comente a flag COMMONFLAGS do buildOptionsForArduino e salve o arquivo.

```
Programmer's Notepad - [config.oil *]
File Edit Search View Tools Window Help
Plain Text
Find
Projects
config.oil *
1 CPU avrArduino {
2
3     BUILD_OPTIONS buildOptionsForArduino {
4         PRELDFLAGS = "-lm"; //math lib required
5         PREASFLAGS = "-x assembler-with-cpp";
6         //COMMONFLAGS = "-Wno-unused-but-set-variable";
7     }
```

Uma vez alterada as configurações do buildOptionsForArduino, você pode proceder com a execução do script **make.py**

## 4.4 GRAVAÇÃO DE UM PROGRAMA COMPILADO PARA ARDUINO UNO

Nas etapas anteriores realizamos a compilação e a geração de uma imagem binária (arquivo trampuinoBlink.hex). Isto significa que estamos prontos para realizar a gravação da imagem no target. Para tanto, utilizaremos a ferramenta **arduino-cli** para tal finalidade.

**Para o target Arduino UNO:**

```
> arduino-cli upload -b arduino:avr:uno -p COM25 -v -i trampuinoBlink.hex -t
```

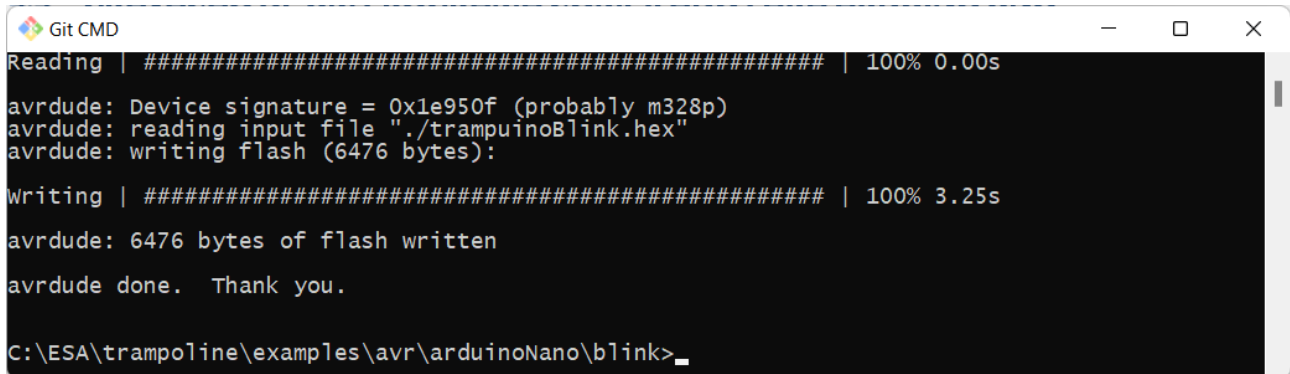
**Para o target Arduino NANO (Old BootLoader):**

```
> arduino-cli upload -b arduino:avr:nano:cpu=atmega328old -p COM25 -v -i trampuinoBlink.hex -t
```

### Para o target Arduino NANO:

```
> arduino-cli upload -b arduino:avr:nano -p COM25 -v -i trampuinoBlink.hex -t
```

Caso a mensagem: **avrdude done. Thank you.** apareça ao final da execução do programa, isto significa que a gravação da imagem no target ocorreu com sucesso.



```
Git CMD
Reading | ##### | 100% 0.00s
avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: reading input file "./trampuinoBlink.hex"
avrdude: writing flash (6476 bytes):

Writing | ##### | 100% 3.25s
avrdude: 6476 bytes of flash written
avrdude done. Thank you.

C:\ESA\trampoline\examples\avr\arduinoNano\blink>_
```

## 4.5 AUTOMAÇÃO UTILIZANDO MACROS

Para automatizar o processo de compilação, geração e download de firmware no Arduino, você deve criar cinco macros seguindo os mesmos passos descritos na seção 1.2. Para tanto, abaixo estão ilustrados os cinco arquivos scripts para automatizar o processo.

### Arquivo trampoline\_runcoil.scp

```
//[Trampoline] Run Goil
echo *****
echo ** Universidade de Brasilia - UnB
echo ** Faculdade do Gama
echo ** Curso: Engenharia Automotiva
echo ** Disciplina: Engenharia de Software Automotivo
echo *****
echo *****
echo ** [Trampoline] Run goil - Oil compiler for $(NAME_PART).oil
echo *****
NPP_SAVEALL
cd $(CURRENT_DIRECTORY)
INPUTBOX "Select the arduino board (options: nano, nano_old or uno):"Board": "$(board)"
set BOARD = $(INPUT)
if "$(BOARD)" == "nano_old" then
    set TARGET_BOARD = nano
    set FQBN = arduino:avr:nano:cpu=atmega328old
else if "$(BOARD)" == "nano" then
    set TARGET_BOARD = nano
    set FQBN = arduino:avr:nano
    set BUILD_DIR = arduino.avr.nano
```

```

else if "$(BOARD)" == "uno" then
    set TARGET_BOARD = uno
    set BUILD_DIR = arduino.avr.uno
endif
INPUTBOX "Set relative TRAMPOLINE_BASE_PATH (ex. ../../.):"BASE_PATH":
    "$(trampoline_base_path)"
set TRAMPOLINE_BASE_PATH = $(INPUT)

goil --target=avr/arduino/$(TARGET_BOARD)
    --templates=$(TRAMPOLINE_BASE_PATH)/goil/templates/ $(NAME_PART).oil
echo *****

```

### Arquivo trampoline\_make\_clean.scp

```

//[Trampoline] Make clean
echo *****
echo ** Universidade de Brasilia - UnB
echo ** Faculdade do Gama
echo ** Curso: Engenharia Automotiva
echo ** Disciplina: Engenharia de Software Automotivo
echo *****
echo *****
echo ** [Trampoline] Make clean - clean previous build files
echo *****
NPP_SAVEALL
cmd /c make.py clean

```

### Arquivo trampoline\_make.scp

```

//[Arduino] Compile
NPP_SAVEALL
cmd /c make.py

```

### Arquivo trampoline\_make\_all.scp

```

//[Trampoline] Make all
echo *****
echo ** Universidade de Brasilia - UnB
echo ** Faculdade do Gama
echo ** Curso: Engenharia Automotiva
echo ** Disciplina: Engenharia de Software Automotivo
echo *****
echo *****
echo ** [Trampoline] Make all - Run goil, make clean and make
echo ** in one step
echo *****
NPP_SAVEALL
NPP_EXEC "$(NPP_DIRECTORY)\plugins\NppExec\scripts\trampoline_goil.scp"
NPP_EXEC "$(NPP_DIRECTORY)\plugins\NppExec\scripts\trampoline_make_clean.scp"
NPP_EXEC "$(NPP_DIRECTORY)\plugins\NppExec\scripts\trampoline_make.scp"

```

### Arquivo trampoline\_upload.scp

```

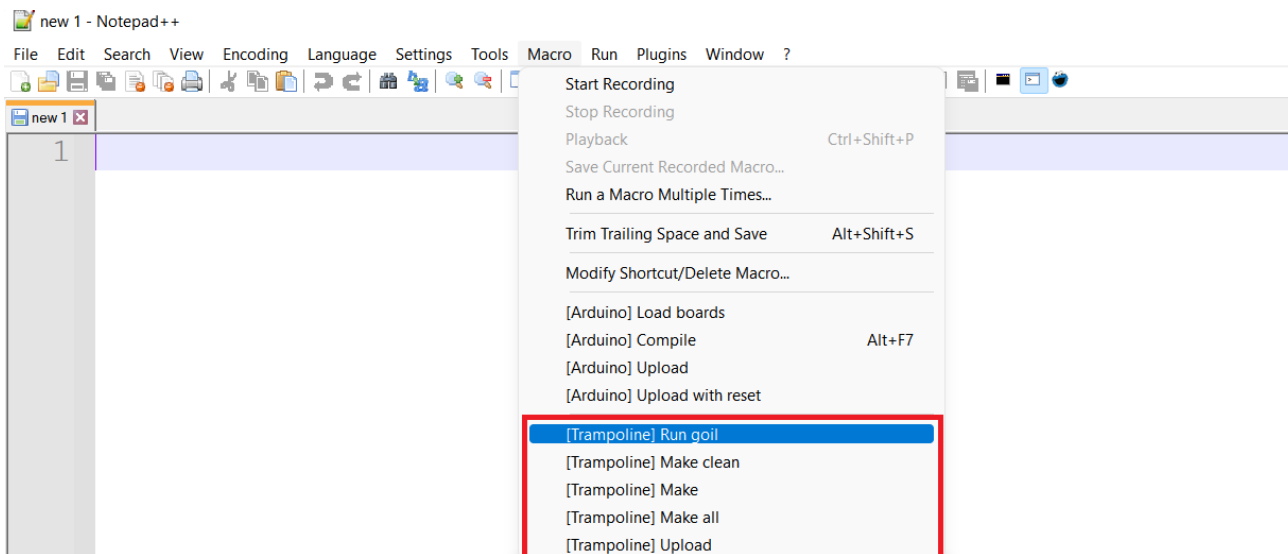
//[Trampoline] Upload

```

```

echo *****
echo ** Universidade de Brasilia - UnB
echo ** Faculdade do Gama
echo ** Curso: Engenharia Automotiva
echo ** Disciplina: Engenharia de Software Automotivo
echo *****
echo *****
echo ** [Trampoline] Upload - upload .hex file to target
echo *****
cd $(CURRENT_DIRECTORY)
INPUTBOX "Select the serial port:":"CommPort": "COM3"
set local commport = $(INPUT)
arduino-cli upload -b $(FQBN) -p $(commport) -v -i image.hex -t

```



## 5 MÓDULO MCP2515\_CAN

O CAN é um protocolo de comunicação serial síncrono de tempo real com alto nível de segurança. Ele é comumente utilizado para integração de ECUs em uma rede de comunicação veicular. O shield MCP2515\_CAN utiliza um controlador CAN MCP2515 que se comunica com um microcontrolado através de uma interface SPI e um transceptor TJA 1050. Utilizando este shield é possível realizar a comunicação entre dois microcontroladores utilizando o protocolo CAN.

Especificações técnicas do shield MCP2515\_CAN:

- Implementa o CAN V2.0B em até 1 Mb/s
- Interface SPI com até 10 MHz
- Frame de dados padrão (11 bits) e estendido (29 bits) e frames remotos
- Dois buffers de recepção com armazenamento prioritário de mensagens
- Tensão de alimentação de 5V
- Resistor de terminação de 120 ohms
- Dois leds indicadores

### 5.1 INSTALAÇÃO DA BIBLIOTECA MCP\_CAN

#### Passo 1 : Execute o programa Git bash

Clique em Botão iniciar -> buscar programas e arquivos - digite: git bash



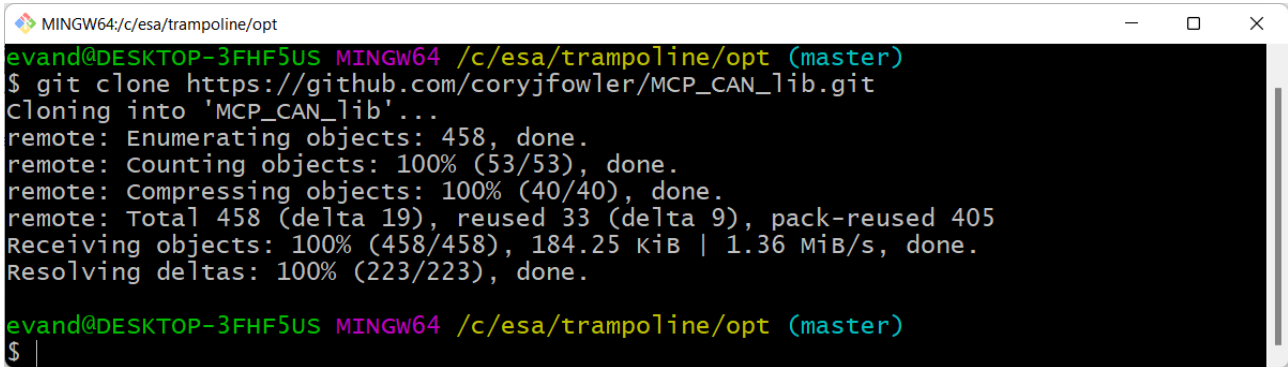
#### Passo 2 : Mudar para o diretório opcional contido no diretório base do Trampoline

```
$ cd trampoline/opt
```

### Passo 3 : Instalação da biblioteca MCP\_CAN

Este comando permite obter a última versão da biblioteca MPC\_CAN. Observe que ao final da execução do comando *git clone*, o usuário poderá verificar (através do comando *ls*) que o diretório CAN\_BUS\_Shield foi criado dentro do diretório opcional.

```
$ git clone https://github.com/coryjfowler/MCP_CAN_lib.git
```

A screenshot of a terminal window titled 'MINGW64:/c/esa/trampoline/opt'. The prompt is 'evand@DESKTOP-3FHF5US MINGW64 /c/esa/trampoline/opt (master)'. The user enters the command '\$ git clone https://github.com/coryjfowler/MCP\_CAN\_lib.git'. The terminal output shows the cloning process: 'Cloning into 'MCP\_CAN\_lib'...', 'remote: Enumerating objects: 458, done.', 'remote: Counting objects: 100% (53/53), done.', 'remote: Compressing objects: 100% (40/40), done.', 'remote: Total 458 (delta 19), reused 33 (delta 9), pack-reused 405', 'Receiving objects: 100% (458/458), 184.25 KiB | 1.36 MiB/s, done.', and 'Resolving deltas: 100% (223/223), done.'. The prompt returns to '\$'.

### Passo 4 : Crie o diretório mcp\_can de forma recursiva no diretório libraries

```
$ mkdir -p ../machines/avr/arduino/libraries/mcp_can/src
```

### Passo 5 : Copie o arquivo de código-fonte e os arquivos cabeçalhos

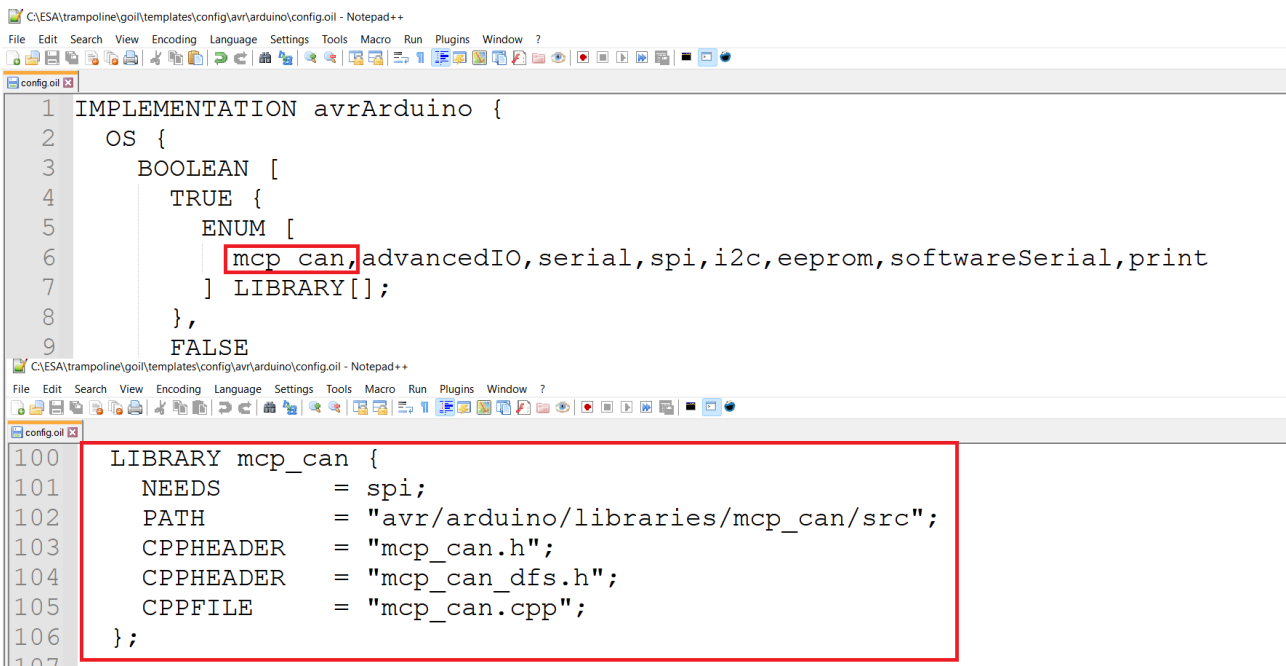
```
$ cp MCP_CAN_lib/mcp_can* ../machines/avr/arduino/libraries/mcp_can/src/
```

## Passo 6 : Declarar a biblioteca MCP\_CAN no template de configuração

Para utilizar efetivamente a biblioteca MCP\_CAN é necessário realizar sua declaração no template de configuração do *target*. Para tanto, abra novamente o arquivo config.oil localizado no seguinte diretório:

```
C:\ESA\trampoline\goil\templates\config\avr\arduino
```

Edite o arquivo config.oil inserindo as diretivas conforme especificado na figura abaixo:



```
1 IMPLEMENTATION avrArduino {
2   OS {
3     BOOLEAN [
4       TRUE {
5         ENUM [
6           mcp_can,advancedIO,serial,spi,i2c,eeprom,softwareSerial,print
7         ] LIBRARY[];
8       },
9       FALSE
100  LIBRARY mcp_can {
101    NEEDS      = spi;
102    PATH       = "avr/arduino/libraries/mcp_can/src";
103    CPPHEADER  = "mcp_can.h";
104    CPPHEADER  = "mcp_can_dfs.h";
105    CPPFILE    = "mcp_can.cpp";
106  };
107 }
```

A partir deste momento, a biblioteca mcp\_can foi instalada com sucesso permitindo com que suas funções possam ser utilizadas em uma aplicação desenvolvida com recursos do OSEK RTOS.

## 5.2 EXEMPLO DE COMUNICAÇÃO ENTRE DUAS ECUS

Neste tutorial utilizaremos dois programas exemplo da própria biblioteca MCP\_CAN para transmitir e receber periodicamente um frame de dados. Naturalmente algumas alterações foram realizadas nos programas exemplos da referida biblioteca para incluir os recursos disponibilizados pelo Trampoline RTOS. Basicamente, os programas exemplo emulam duas ECUs conectadas via rede CAN. A ECU1 é responsável por transmitir um frame de dados de forma periódica (1.000 ms) enquanto que a ECU2 apresenta o frame de dados recebido na interface serial. A figura 4 apresenta esquematicamente o funcionamento dos programas exemplo.

## Passo 7 : Mudar para o diretório base do Trampoline



## Passo 8 : Criar o diretório de desenvolvimento devel

```
$ mkdir -p opt/devel/can_send && opt/devel/can_send
```

O diretório da aplicação consiste em uma pasta criada dentro do diretório de desenvolvimento. Observa-se que no diretório da aplicação são armazenados todos os arquivos necessários para o desenvolvimento, compilação e geração da imagem binária da aplicação. Crie os arquivos `can_send.oil` e `can_send.cpp` e preenchê-los com o conteúdo estabelecido as figuras abaixo.

```
1 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
2 // Universidade de Brasilia (UnB)
3 // Faculdade do Gama
4 // Curso: Engenharia Automotiva
5 // Disciplina : Engenharia de Software Automotivo
6 ////////////////////////////////////////////////////////////////////////
7
8 OIL_VERSION = "2.5" : "test" ;
9
10 CPU test {
11     OS config {
12         STATUS = STANDARD;
13         BUILD = TRUE {
14             TRAMPOLINE_BASE_PATH = "../..../";
15             APP_NAME      = "image";
16             APP_SRC        = "can_send.cpp";
17             CPPCOMPILER    = "avr-g++";
```



```

18     COMPILER      = "avr-gcc";
19     LINKER        = "avr-gcc";
20     ASSEMBLER     = "avr-gcc";
21     COPIER        = "avr-objcopy";
22     SYSTEM        = PYTHON;
23     LIBRARY        = serial ;
24     LIBRARY        = mcp_can;
25 };
26     SYSTEM_CALL = TRUE;
27 };
28
29     APPMODE stdAppmode {};
30
31     ALARM periodicAlarm {
32         COUNTER= SystemCounter;
33         ACTION = ACTIVATETASK {
34             TASK = SendCanFrame;
35         };
36         AUTOSTART = TRUE {
37             ALARMTIME = 1000; //ativado a cada 1024 ms
38             CYCLETIME = 1000; //ativado a cada 1024 ms
39             APPMODE = stdAppmode;
40         };
41     };
42
43     TASK SendCanFrame {
44         PRIORITY = 20;
45         AUTOSTART = FALSE;
46         ACTIVATION = 1;
47         SCHEDULE = FULL;
48         STACKSIZE = 256;
49     };
50 };

```

O arquivo **can\_send.oil** configura o sistema operacional por meio de uma descrição orientada a objetos dos recursos utilizados. Desse modo, todos os objetos de sistemas especificados pelo padrão OSEK e seus relacionamentos devem ser declarados através da linguagem OIL. Na **linha 14** é especificado o diretório base relativo do Trampoline (TRAMPOLINE\_BASE\_PATH). Observa-se que este parâmetro deve ser configurado em função da localização dos arquivos .cpp e .oil. Já a **linha 15** especifica o nome da aplicação compilada. Esta declaração é utilizada para especificar o nome a ser atribuído ao arquivo contendo a imagem binária gerada (**image.hex**) decorrente do processo de compilação. A **linha 16** especifica o nome do arquivo de código-fonte da aplicação. Na **linha 23** é declarada, de forma explícita, a biblioteca mcp\_can. Esta declaração permite que o desenvolvedor utilize as funções da referida biblioteca no desenvolvimento da aplicação.

Na **linha 30** é declarado explicitamente um objeto alarme. Este objeto é comumente

utilizado para disparar ações periódicas. Na **linha 32** é declarada a ação que o alarme deve executar todas as vezes em que o período do alarme expirar (neste caso, ativar uma task - ACTIVATETASK). A **linha 33** especifica a task que deverá ser ativada todas as vezes que o período do alarme expirar. Já na **linha 42** é declarada a task (SendCanFrame) responsável por enviar periodicamente (isto é, a cada ativação do alarme) um frame de dados através da rede CAN.

## Arquivo can\_send.cpp

```

1 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
2 // Universidade de Brasilia (UnB)
3 // Faculdade do Gama
4 // Curso: Engenharia Automotiva
5 // Disciplina : Engenharia de Software Automotivo
6 //////////////////////////////////////////////////////////////////////
7
8 #include "tpl_os.h"
9 #include "Arduino.h"
10 #include "Board.h"
11
12 //Macro para definicoes
13 #define mEEEC1_ID    0x0CF00400
14 #define mEEEC1_DLC   8
15 #define mEEEC1_EXT_FRAME  1
16
17 //Variavel que armazena o FRAME_DATA
18 unsigned char mEEEC1_data[8] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
19
20 //Constroi um objeto MCP_CAN e configura o chip selector para o pino 10.
21 MCP_CAN CAN1(CAN1_CS);
22
23 void setup()
24 {
25     // Inicializa a interface serial : baudrate = 115200
26     Serial.begin(115200);
27
28     // Inicializa o controlador can : baudrate = 250K, clock=8MHz
29     while(CAN1.begin(MCP_ANY, CAN_500KBPS, MCP_8MHZ) != CAN_OK)
30     {
31         delay(200);
32     }
33     Serial.println("MCP2515 can_send inicializado com sucesso!");
34     //Modifica para o modo normal de operação
35     CAN1.setMode(MCP_NORMAL);
36 }
37 //////////////////////////////////////////////////////////////////////
38 // Task SendCanFrame
39 // Utilizada enviar as mensagens CAN

```



```

4 // Curso: Engenharia Automotiva
5 // Disciplina : Engenharia de Software Automotivo
6 ///////////////////////////////////////////////////////////////////
7
8 OIL_VERSION = "2.5" : "test" ;
9
10 CPU test {
11     OS config {
12         STATUS = STANDARD;
13         BUILD = TRUE {
14             TRAMPOLINE_BASE_PATH = ".././../";
15             APP_NAME      = "image";
16             APP_SRC       = "can_receive.cpp";
17             CPPCOMPILER   = "avr-g++";
18             COMPILER      = "avr-gcc";
19             LINKER        = "avr-gcc";
20             ASSEMBLER     = "avr-gcc";
21             COPIER        = "avr-objcopy";
22             SYSTEM        = PYTHON;
23             LIBRARY       = serial ;
24             LIBRARY       = mcp_can;
25         };
26         SYSTEM_CALL = TRUE;
27     };
28
29     APPMODE stdAppmode {};
30
31     ALARM periodicAlarm {
32         COUNTER= SystemCounter;
33         ACTION = ACTIVATETASK {
34             TASK = ReceiveCanFrame;
35         };
36         AUTOSTART = TRUE {
37             ALARMTIME = 500; //ativado a cada 512 ms
38             CYCLETIME = 500; //ativado a cada 512 ms
39             APPMODE  = stdAppmode;
40         };
41     };
42
43     TASK ReceiveCanFrame {
44         PRIORITY  = 20;
45         AUTOSTART = FALSE;
46         ACTIVATION = 1;
47         SCHEDULE  = FULL;
48         STACKSIZE = 256;
49     };
50 };

```

A **linha 15** especifica o nome base da aplicação compilada. A **linha 16** especifica o nome do arquivo de código-fonte da aplicação. Na **linha 23** é declarada, de forma explícita,

a biblioteca `mcp_can`. Como está prevista a utilização da interface serial na aplicação é necessário declarar a biblioteca serial através da instrução estabelecida na **linha 24**.

Na **linha 30** é declarado explicitamente um objeto alarme. Na **linha 32** é declarada a ação que o alarme deve executar todas as vezes em que o período do alarme expirar (neste caso, ativar uma task - ACTIVATETASK). A ação consiste em ativar a task ReceiveCanFrame (**linha 34**). Para garantir que não haja perda de um frame de dados, o alarme foi configurado com o período de 500ms (**linhas 37 e 38**). Na **linha 43** é declarada a task (ReceiveCanFrame) responsável por verificar periodicamente a recepção de frame de dados.

## Arquivo can\_receive.cpp

```
1 //////////////////////////////////////  
2 // Universidade de Brasilia (UnB)  
3 // Faculdade do Gama  
4 // Curso: Engenharia Automotiva  
5 // Disciplina : Engenharia de Software Automotivo  
6 //////////////////////////////////////  
7  
8 #include "tpl_os.h"  
9 #include "Arduino.h"  
10 #include "Board.h"  
11  
12 //Variavel para armazenar informacoes do frame recebido  
13 unsigned char    mDLC = 0;  
14 unsigned char    mData[8];  
15 long unsigned int   mID;  
16  
17 //Constroi um objeto MCP_CAN e configura o chip selector para o pino 10.  
18 MCP_CAN CAN1(CAN1_CS);  
19  
20 void setup()  
21 {  
22     // Inicializa a interface serial : baudrate = 115200  
23     Serial.begin(115200);  
24  
25     // Inicializa o controlador can : baudrate = 500K, clock=08MHz  
26     while(CAN1.begin(MCP_ANY, CAN_500KBPS, MCP_8MHZ) != CAN_OK)  
27         {  
28             delay(200);  
29         }  
30     Serial.println("Modulo CAN inicializado!");  
31     //Configura o controlador MCP2515 no modo normal.  
32     CAN1.setMode(MCP_NORMAL);  
33     //Configura o pino de interrupção para recepção  
34     pinMode(CAN_INT, INPUT);  
35     Serial.println("MCP2515 exemplo can_receive...");  
36     Serial.print("ID\tType\DLC\tByte0\tByte1\tByte2");  
37     Serial.println("\tByte3\tByte4\tByte5\tByte6\tByte7");
```

```

38 }
39 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
40 // Task ReceiveCanFrame
41 // Utilizada receber as mensagens CAN
42 // Período de execução da task: 500ms
43 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
44 TASK(ReceiveCanFrame)
45 {
46     //Se uma interrupção ocorreu interrupção (CAN_INT pino = 0), lê o buffer de recepção
47     if (!digitalRead(CAN_INT))
48     {
49         //Lê os dados: mID = identificador, mDLC = comprimento, mData = dados do
           frame
50         CAN1.readMsgBuf(&mID, &mDLC, mData);
51
52         //Determina se o frame é do tipo standard (11 bits) ou estendido (29 bits)
53         if ((mID & CAN_IS_EXTENDED) == CAN_IS_EXTENDED)
54         {
55             sprintf(msgString, "0x%.8lX\t\tExt\t[%1d]\t", (mID & CAN_EXTENDED_ID),
               mDLC);
56         }
57         else
58         {
59             sprintf(msgString, "0x%.3lX\t\tStd\t[%1d]\t", mID, mDLC);
60         }
61
62         Serial.print(msgString);
63         //Verifica se a mensagem é do tipo de requisição remota.
64         if ((mID & CAN_IS_REMOTE_REQUEST) == CAN_IS_REMOTE_REQUEST)
65         {
66             sprintf(msgString, "rrf");
67             Serial.print(msgString);
68         }
69         else
70         {
71             for(byte i = 0; i < mDLC; i++)
72             {
73                 sprintf(msgString, "0x%.2X\t", mData[i]);
74                 Serial.print(msgString);
75             }
76         }
77         Serial.println();
78     }
79 }

```

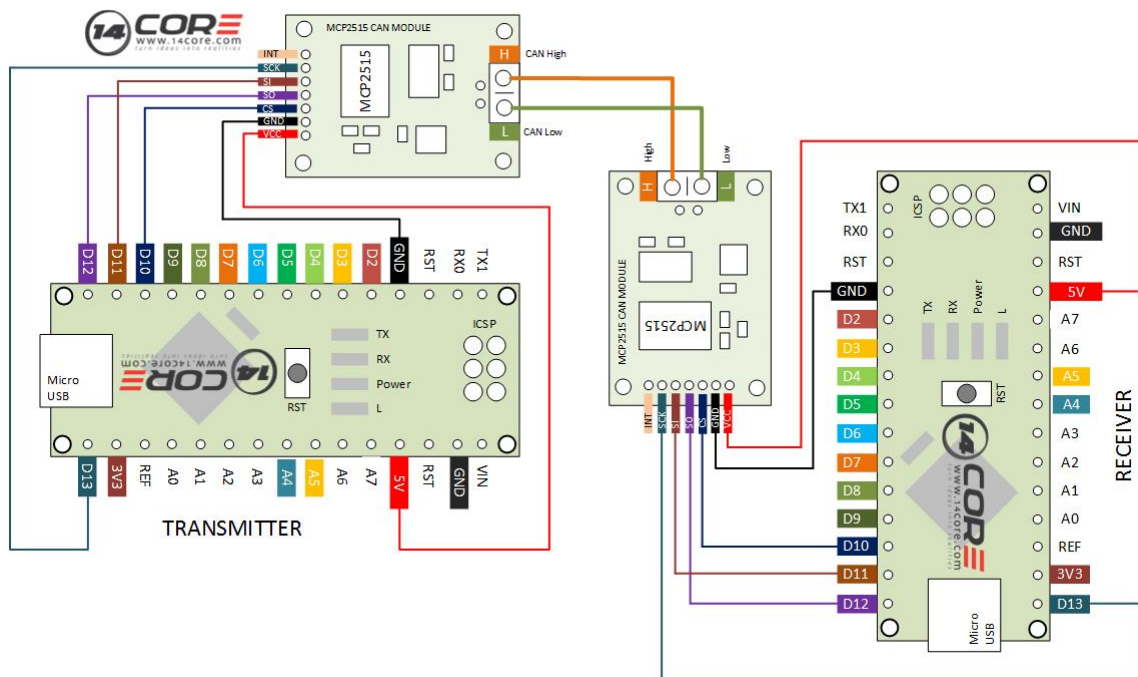
O arquivo `can_receive.cpp`, também desenvolvido com base em um arquivo exemplo da biblioteca `mcp_can`, é responsável por receber as mensagens (frames) recebidas através da rede CAN e enviá-las para a interface serial do Arduino. A **linha 47** verifica se alguma mensagem foi recebida no buffer de recepção do controlador CAN. Caso alguma mensagem

tenha sido recebida, as instruções contidas nas **linhas 49 a 82** poderão ser executadas. A **linha 54** copia o comprimento da mensagem (campo de dados) para a variável mDLC e os dados contidos no buffer de recepção para a variável mData. A **linha 57** executa ação semelhante, todavia, o ID da mensagem recebida. As **linhas 60 a 82** são responsáveis somente pela impressão da mensagem na interface serial.

### 5.3 CONEXÃO E COMPILAÇÃO DOS PROGRAMAS EXEMPLO

Para testar a comunicação entre dois dispositivos via rede CAN é necessário realizar a conexão física entre dois Arduinos e dois módulos MCP2515 conforme especificado na figura abaixo. Uma vez realizada a conexão física entre dois Arduinos via rede CAN, você deve realizar a compilação dos programas exemplos apresentados na seção 4.3. Para compilação e gravação siga os passos descritos nas seções 4.3 e 4.4 respectivamente.

**FIGURA 5: CONEXÃO FÍSICA ENTRE DOIS ARDUINOS VIA REDE CAN**



**FIGURA 6: EXECUÇÃO DOS PROGRAMAS EXEMPLO DE COMUNICAÇÃO**

COM5

```
Enter setting mode success
set rate success!!
Enter Normal Mode Success!!
can_send: mensagem transmitida com sucesso
can_send: mensagem transmitida com sucesso
can_send: mensagem transmitida com sucesso
can_send: mensagem transmitida com sucesso
can_send: mensagem transmitida com sucesso
can_send: mensagem transmitida com sucesso
can_send: mensagem transmitida com sucesso
can_send: mensagem transmitida com sucesso
```

COM6

```
Enter setting mode success
set rate success!!
Enter Normal Mode Success!!
Modulo CAN inicializado!
Time      PGN      Byte0  Byte1  Byte2  Byte3
1024      0xCF00400  0xFF  0xFF  0xFF  0xFF
2048      0xCF00400  0xFF  0xFF  0xFF  0xFF
3072      0xCF00400  0xFF  0xFF  0xFF  0xFF
4096      0xCF00400  0xFF  0xFF  0xFF  0xFF
5120      0xCF00400  0xFF  0xFF  0xFF  0xFF
6144      0xCF00400  0xFF  0xFF  0xFF  0xFF
7168      0xCF00400  0xFF  0xFF  0xFF  0xFF
8192      0xCF00400  0xFF  0xFF  0xFF  0xFF
```