robocadSim

Release 1.3.7

crackanddie

Mar 15, 2022

GETTING STARTED

Welcome to the robocadSim documentation page. Here you will find lots of information about how to program robocadSim' virtual robots.

CHAPTER

ONE

GETTING STARTED

Where to begin?

First you need to download robocadSim and run it. Latest version.

I hope you have already chosen the language with which you will program your robot and the IDE for it.

You can read how to start using libraries for the selected language here.

Write a program, turn on the robot and run the code!

CHAPTER

TWO

ROBOT DOCUMENTATION

Here You can choose the robot you are programming. Read the documentation about all functions in the selected language and see some helpful examples. Good luck!

2.1 RE21

Some info here

2.2 RE21mini

Everything about RE21mini

2.2.1 Functions for RE21mini

Motor Control

Use motor control variables to set speeds to motors.

Python

C#

Location and name:

- RobocadSim.RE21mini.right_motor_speed
- RobocadSim.RE21mini.left_motor_speed
- RobocadSim.RE21mini.back_motor_speed

Set:

- *float* speed to right motor
- *float* speed to left motor
- *float* speed to back motor

Get:

Example:

```python
from robocadSimPy import RobocadSim


robot = RobocadSim.RE21mini()
robot.connect()
robot.right_motor_speed = 10
robot.left_motor_speed = -10
robot.back_motor_speed = 0
robot.disconnect()
```

Additional info:

- Range of speed is from -50 to 50

Location and name:

- RobocadSim.RE21mini.rightMotorSpeed
- RobocadSim.RE21mini.leftMotorSpeed
- RobocadSim.RE21mini.backMotorSpeed

Set:

- *float* speed to right motor
- *float* speed to left motor
- *float* speed to back motor

Get:

Example:

```csharp
using System;
using RobocadSim;

namespace TestLib
{
    class Program
    {
        static void Main(string[] args)
        {
            RE21mini robot = new RE21mini();
            robot.Connect();
            robot.rightMotorSpeed = 10;
            robot.leftMotorSpeed = -10;
            robot.backMotorSpeed = 0;
            robot.Disconnect();
        }
    }
}
```

Additional info:

- Range of speed is from -50 to 50

OMS Control

Use OMS control variables to set speeds, angles and directions to motors and servo motors.

Python

C#

Location and name:

- RobocadSim.RE21mini.lift_motor_speed
- RobocadSim.RE21mini.angle_to_big
- RobocadSim.RE21mini.dir_to_plates

Set:

- *float* speed to lift motor
- *float* angle to big servo
- *float* direction to small servo

Get:

Example:

```
from robocadSimPy import RobocadSim


robot = RobocadSim.RE21mini()
robot.connect()
robot.lift_motor_speed = 10
robot.angle_to_big = 1600
robot.dir_to_plates = 1400
robot.disconnect()
```

Additional info:

- Range of speed is from -50 to 50
- Range of angles for big servo motor is from 1490 to 1750
- Range of values for small servo motor is from 1400 to 1600

Location and name:

- RobocadSim.RE21mini.liftMotorSpeed
- RobocadSim.RE21mini.angleToBig
- RobocadSim.RE21mini.dirToPlates

Set:

- *float* speed to lift motor
- *float* angle to big servo
- *float* direction to small servo

Get:

Example:

```
using System;
using RobocadSim;

namespace TestLib
{
    class Program
    {
        static void Main(string[] args)
        {
            RE21mini robot = new RE21mini();
            robot.Connect();
            robot.liftMotorSpeed = 10;
            robot.angleToBig = 1600;
            robot.dirToPlates = 1400;
            robot.Disconnect();
        }
    }
}
```

Additional info:

- Range of speed is from -50 to 50
- Range of angles for big servo motor is from 1490 to 1750
- Range of values for small servo motor is from 1400 to 1600

Reset Control

Use reset control variables to reset to motors' encoders.

Python

C#

Location and name:

- RobocadSim.RE21mini.reset_right_motor_enc
- RobocadSim.RE21mini.reset_left_motor_enc
- RobocadSim.RE21mini.reset_back_motor_enc
- RobocadSim.RE21mini.reset_lift_motor_enc
- RobocadSim.RE21mini.reset_gyro

Set:

- *bool* reset right motor encoder
- *bool* reset left motor encoder
- *bool* reset back motor encoder
- *bool* reset lift motor encoder
- *bool* reset gyroscope

Get:

Example:

```
from robocadSimPy import RobocadSim


robot = RobocadSim.RE21mini()
robot.connect()
robot.reset_right_motor_enc = True
robot.reset_left_motor_enc = True
robot.reset_back_motor_enc = True
robot.reset_lift_motor_enc = True
robot.reset_gyro = False
robot.disconnect()
```

Additional info:

- You should write Your own gyro reset (cause my doesn't work well :))

Location and name:

- RobocadSim.RE21mini.resetRightMotorEnc
- RobocadSim.RE21mini.resetLeftMotorEnc
- RobocadSim.RE21mini.resetBackMotorEnc
- RobocadSim.RE21mini.resetLiftMotorEnc
- RobocadSim.RE21mini.resetGyro

Set:

- *bool* reset right motor encoder
- *bool* reset left motor encoder
- *bool* reset back motor encoder
- *bool* reset lift motor encoder
- *bool* reset gyroscope

Get:

Example:

```
using System;
using RobocadSim;

namespace TestLib
{
    class Program
    {
        static void Main(string[] args)
        {
            RE21mini robot = new RE21mini();
            robot.Connect();
            robot.resetRightMotorEnc = true;
            robot.resetLeftMotorEnc = true;
            robot.resetBackMotorEnc = true;
            robot.resetLiftMotorEnc = true;
            robot.resetGyro = false;
```

(continues on next page)

(continued from previous page)

```
            robot.Disconnect();
        }
    }
}
```

Additional info:

- You should write Your own gyro reset (cause my doesn't work well :))

Encoder Indicator

Use encoder indicator variables to get motors' encoders' values.

Python

C#

Location and name:

- RobocadSim.RE21mini.right_motor_enc
- RobocadSim.RE21mini.left_motor_enc
- RobocadSim.RE21mini.back_motor_enc
- RobocadSim.RE21mini.lift_motor_enc

Set:

Get:

- *float* right motor' encoder' value
- *float* right motor' encoder' value
- *float* right motor' encoder' value
- *float* right motor' encoder' value

Example:

```
from robocadSimPy import RobocadSim


robot = RobocadSim.RE21mini()
robot.connect()
encoders = [0] * 4
encoders[0] = robot.right_motor_enc
encoders[1] = robot.left_motor_enc
encoders[2] = robot.back_motor_enc
encoders[3] = robot.lift_motor_enc
robot.disconnect()
```

Additional info:

- You should use Transfunction with encoders for a more convenient representation of values

Location and name:

- RobocadSim.RE21mini.rightMotorEnc

- RobocadSim.RE21mini.leftMotorEnc
- RobocadSim.RE21mini.backMotorEnc
- RobocadSim.RE21mini.liftMotorEnc

Set:

Get:

- *float* right motor' encoder' value
- *float* right motor' encoder' value
- *float* right motor' encoder' value
- *float* right motor' encoder' value

Example:

```
using System;
using RobocadSim;

namespace TestLib
{
    class Program
    {
        static void Main(string[] args)
        {
            RE21mini robot = new RE21mini();
            robot.Connect();
            float[] encoders = new float[4];
            encoders[0] = robot.rightMotorEnc;
            encoders[1] = robot.leftMotorEnc;
            encoders[2] = robot.backMotorEnc;
            encoders[3] = robot.liftMotorEnc;
            robot.Disconnect();
        }
    }
}
```

Additional info:

- You should use Transfunction with encoders for a more convenient representation of values

Button Indicator

Use button indicator variables to get buttons states.

Python

C#

Location and name:

- RobocadSim.RE21mini.button_ems
- RobocadSim.RE21mini.button_start
- RobocadSim.RE21mini.button_limit

Set:

Get:

- *bool* EMS button state
- *bool* Start button state
- *bool* Limit button state

Example:

```
from robocadSimPy import RobocadSim


robot = RobocadSim.RE21mini()
robot.connect()
buttons = [0] * 3
buttons[0] = robot.button_ems
buttons[1] = robot.button_start
buttons[2] = robot.button_limit
robot.disconnect()
```

Additional info:

Location and name:

- RobocadSim.RE21mini.buttonEMS
- RobocadSim.RE21mini.buttonStart
- RobocadSim.RE21mini.buttonLimit

Set:

Get:

- *bool* EMS button state
- *bool* Start button state
- *bool* Limit button state

Example:

```
using System;
using RobocadSim;

namespace TestLib
{
    class Program
    {
        static void Main(string[] args)
        {
            RE21mini robot = new RE21mini();
            robot.Connect();
            bool[] buttons = new bool[3];
            buttons[0] = robot.buttonEMS;
```

(continues on next page)

(continued from previous page)

```
            buttons[1] = robot.buttonStart;
            buttons[2] = robot.buttonLimit;
            robot.Disconnect();
        }
    }
}
```

Additional info:

Sensor Indicator

Use sensor indicator variables to get sensors values.

Python

C#

Location and name:

- RobocadSim.RE21mini.right_us
- RobocadSim.RE21mini.left_us
- RobocadSim.RE21mini.right_ir
- RobocadSim.RE21mini.left_ir
- RobocadSim.RE21mini.gyro

Set:

Get:

- *float* right ultrasonic sensor value
- *float* left ultrasonic sensor value
- *float* right infrared sensor value
- *float* left infrared sensor value
- *float* gyro value

Example:

```
from robocadSimPy import RobocadSim


robot = RobocadSim.RE21mini()
robot.connect()
sensors = [0] * 5
sensors[0] = robot.right_us
sensors[1] = robot.left_us
sensors[2] = robot.right_ir
sensors[3] = robot.left_ir
sensors[4] = robot.gyro
robot.disconnect()
```

Additional info:

Location and name:

- RobocadSim.RE21mini.rightUS
- RobocadSim.RE21mini.leftUS
- RobocadSim.RE21mini.rightIR
- RobocadSim.RE21mini.leftIR
- RobocadSim.RE21mini.gyro

Set:

Get:

- *float* right ultrasonic sensor value
- *float* left ultrasonic sensor value
- *float* right infrared sensor value
- *float* left infrared sensor value
- *float* gyro value

Example:

```
using System;
using RobocadSim;

namespace TestLib
{
    class Program
    {
        static void Main(string[] args)
        {
            RE21mini robot = new RE21mini();
            robot.Connect();
            float[] sensors = new float[5];
            sensors[0] = robot.rightUS;
            sensors[1] = robot.leftUS;
            sensors[2] = robot.rightIR;
            sensors[3] = robot.leftIR;
            sensors[4] = robot.gyro;
            robot.Disconnect();
        }
    }
}
```

Additional info:

CHAPTER

THREE

FIELD DOCUMENTATION

Here You can choose a field and read some info about it.

3.1 Field E21

Here You can read some info about E21 field.

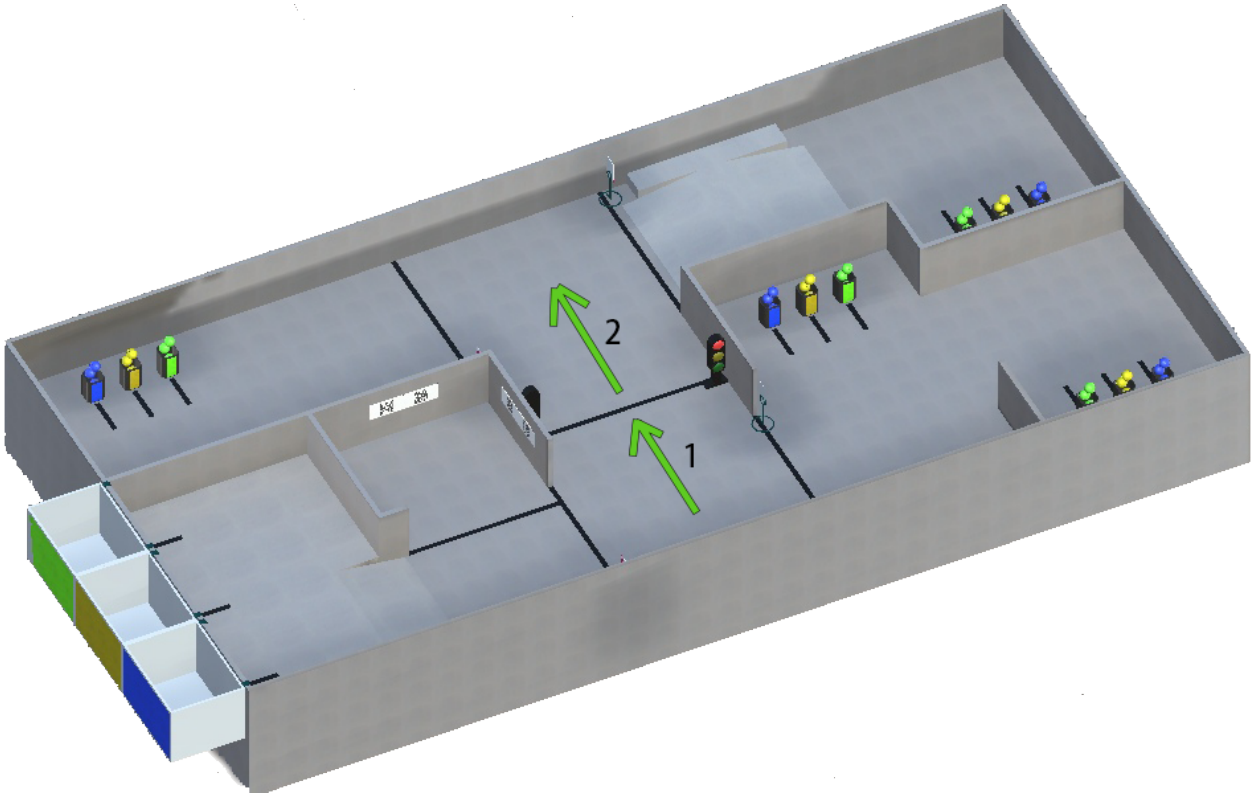
3.1.1 Traffic lights on E21 field

There are 2 traffic lights on the field and they can burn with such colors:

- **Red** color
- **Yellow** color
- **Green** color

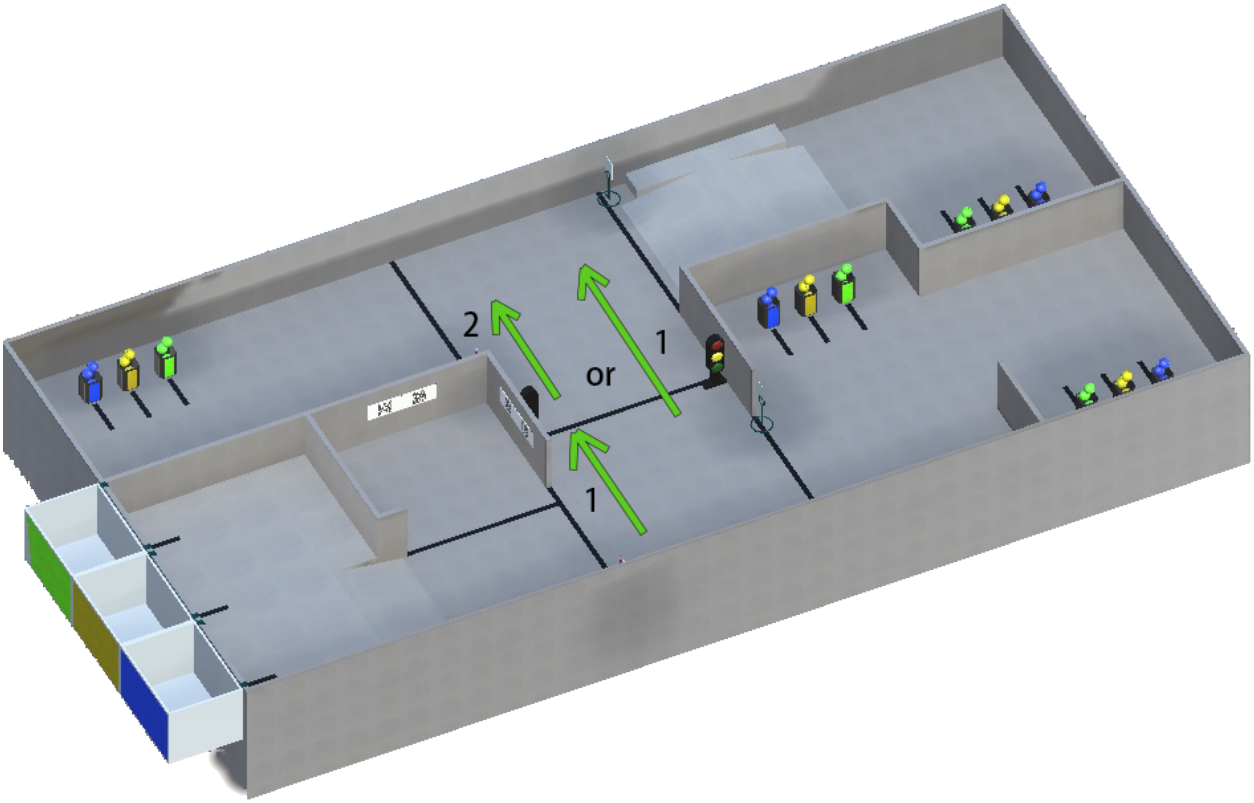
Red:

If a **red** light is on at a traffic light, robot should stop in front of it or in front of a black line. And continue movement when a **green** light is on at a traffic light:



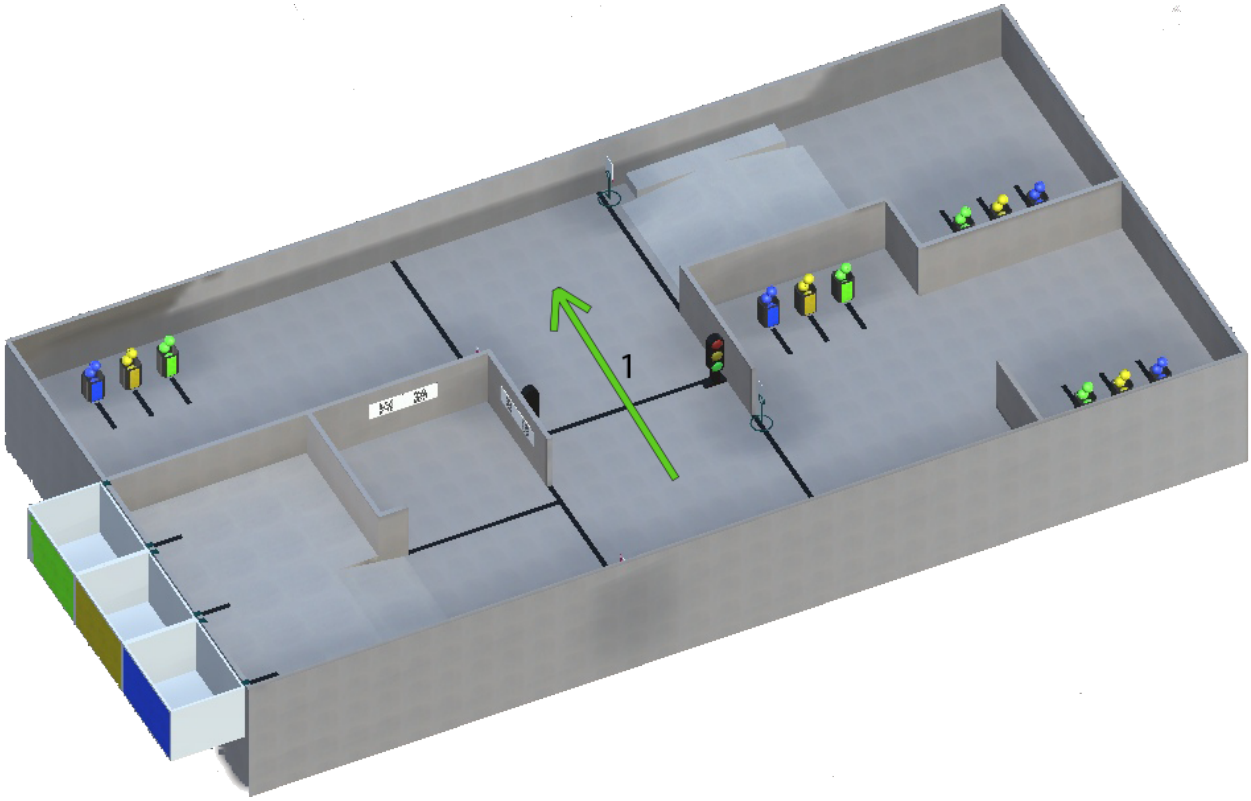
Yellow:

If a **yellow** light is on at a traffic light, robot should stop in front of it or in front of a black line. And continue movement when a **green** light is on at a traffic light. **But** if robot already crossed the line when the yellow light came on, it can complete movement:



Green:

If a **green** light is on at a traffic light, robot can continue its movement:



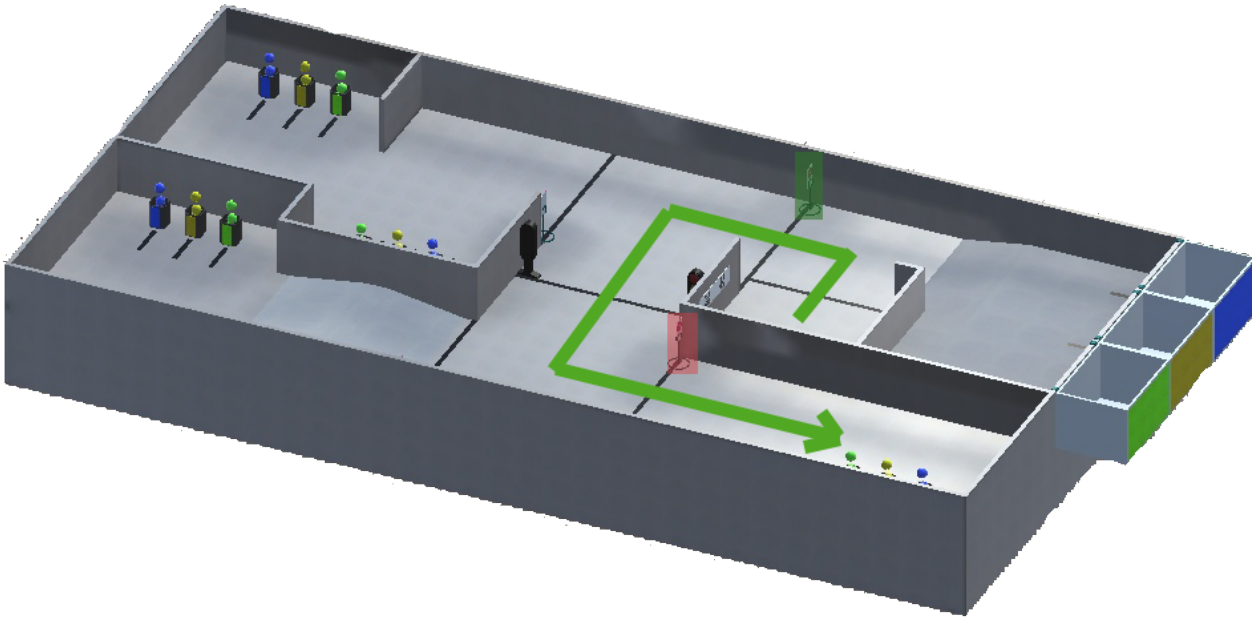
3.1.2 Signs on E21 field

There are 4 signs on the field and they may be:

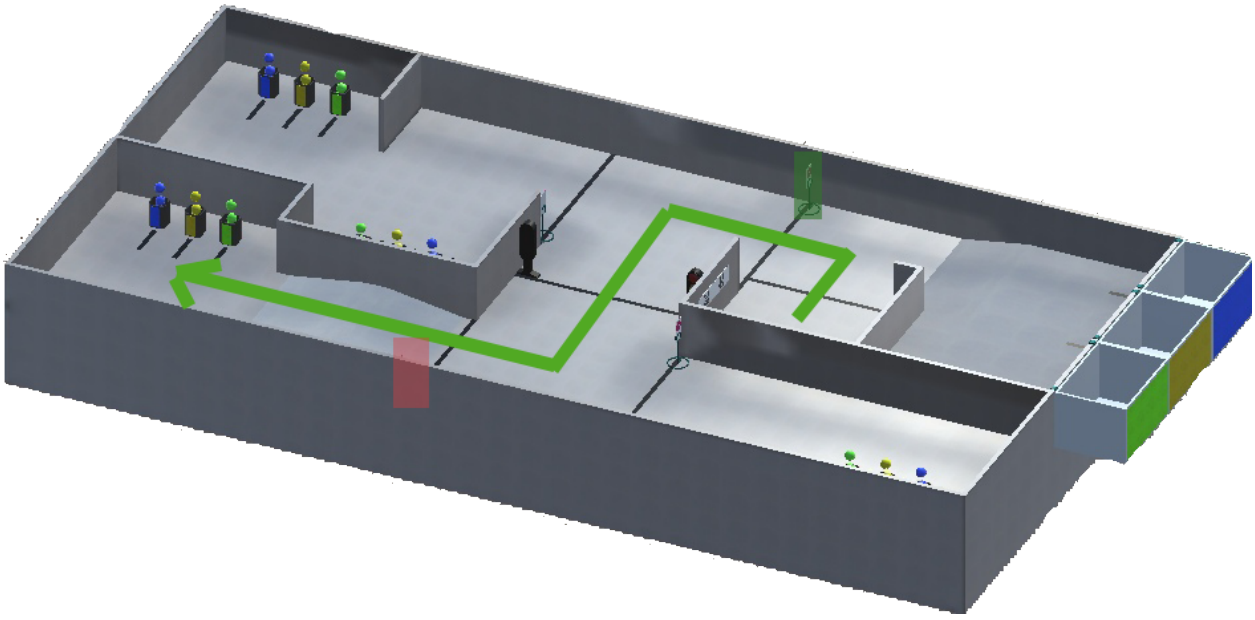
- **Turn Left** sign
- **Turn Right** sign
- **Move Forward** sign
- **Stop** sign

Turn Left:

For example, if You have **Turn Left** sign at the start (green rect) You should move like that:



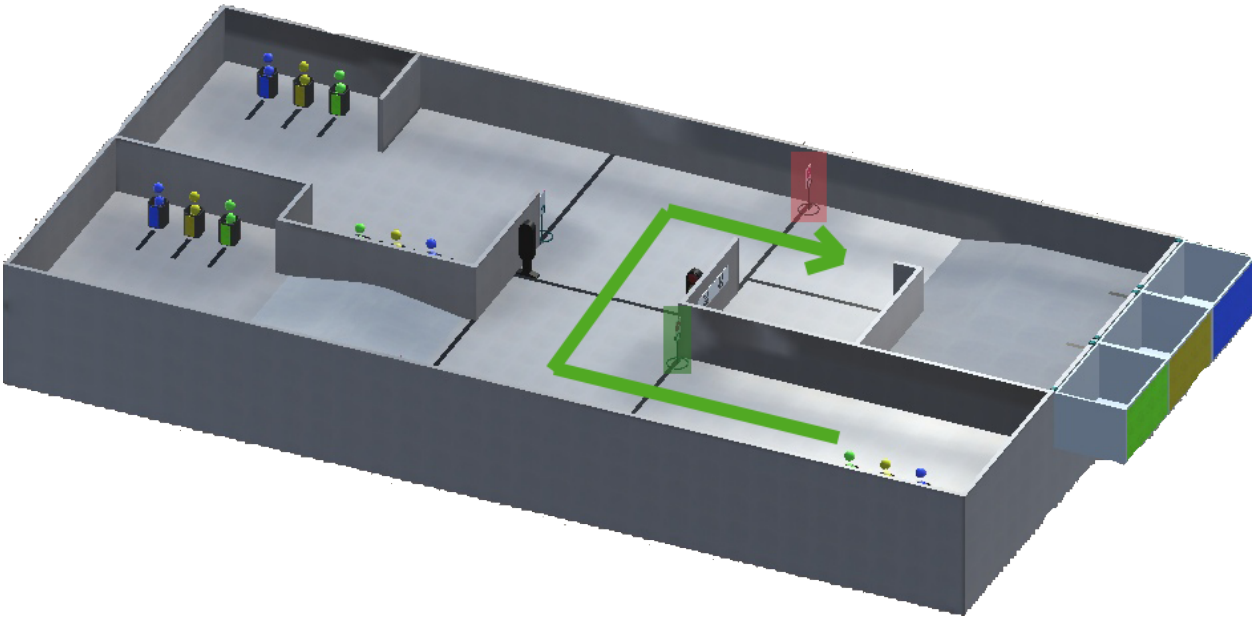
or like that:



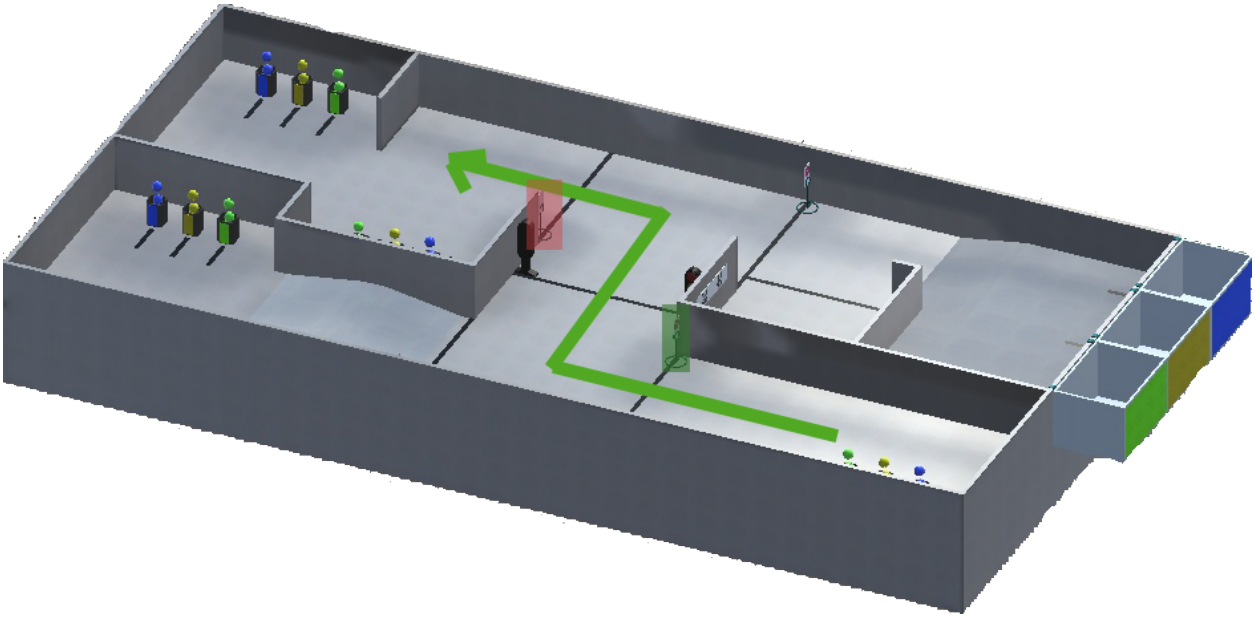
and check next sign (red rect) if You need it.

Turn Right:

For example, if there is **Turn Right** sign at the Distant village (green rect) You should move like that:



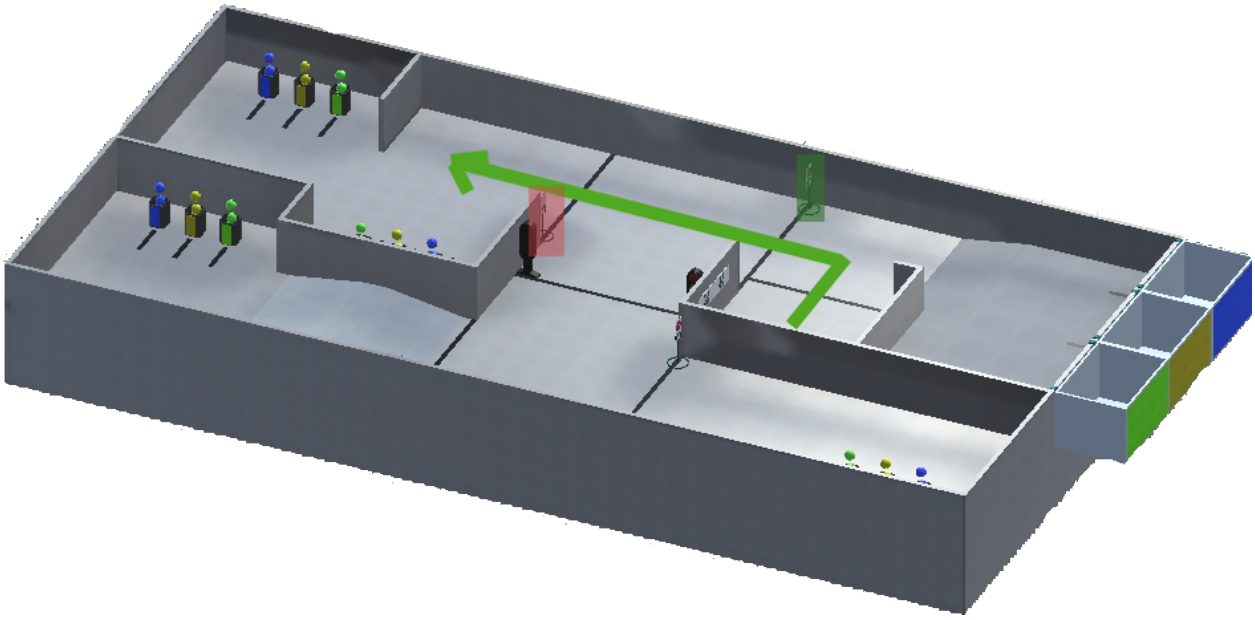
or like that:



and check next sign (red rect) if You need it.

Move Forward:

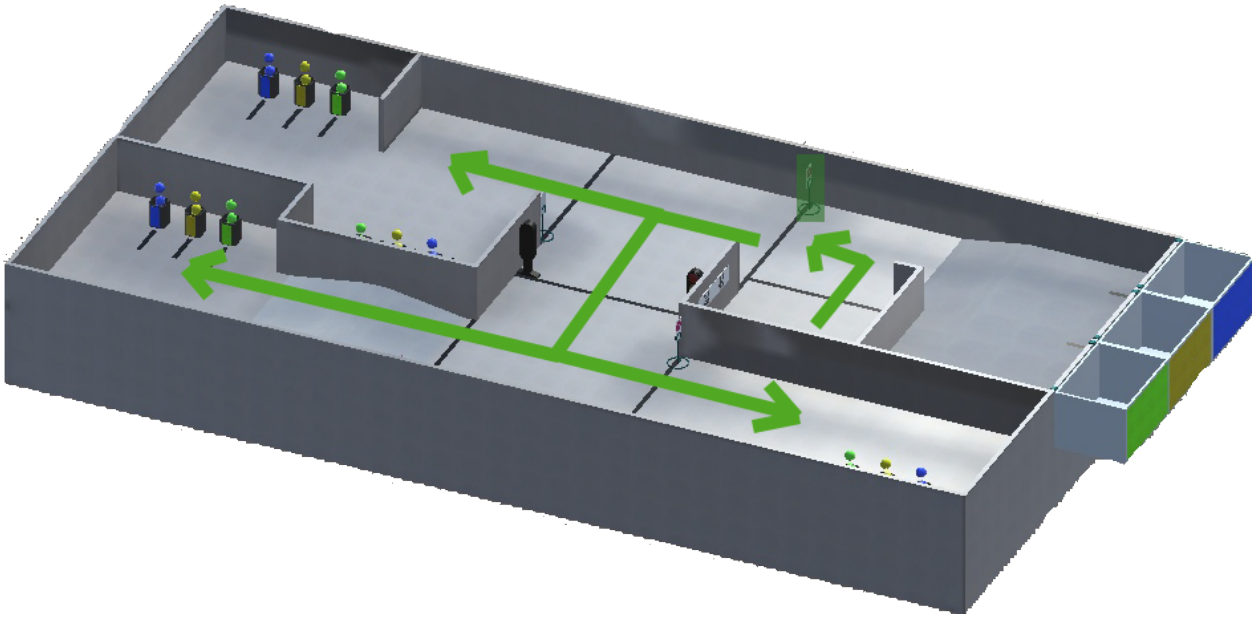
For example, if there is **Move Forward** sign at the start (green rect) You should move like that:



and check next sign (red rect) if You need it.

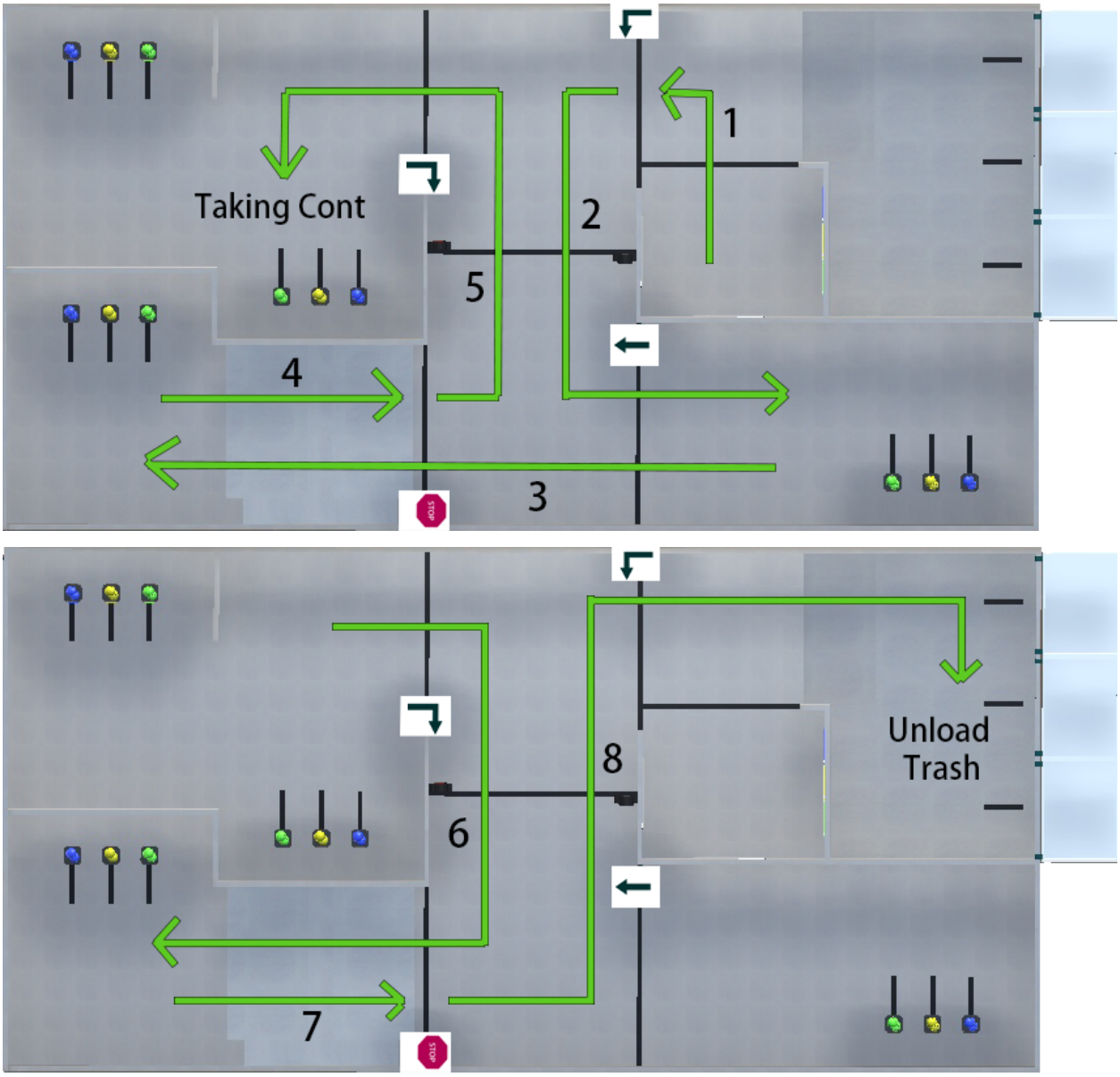
Stop:

For example, if there is **Stop** sign at the start (green rect) You should stop in front of the sign (or line) and wait for 2 seconds. After that You can continue Your movement wherever You want. Example:



All signs:

Here is an example of movement when robot has to go to the Pine village and take containers from there:



CHAPTER

FOUR

FUNCAD DOCUMENTATION

Here You can find some info about Funcad.

4.1 FromAxisToMotors

FromAxisToMotors function is used to remake input axis values into motors values for **tricycle** robot.

Python

C++

C#

Location and name: Funcad.Funcad.from_axis_to_motors()

Inputs:

- *float* speed to X axis
- *float* speed to Y axis
- *float* speed to Z axis

Output:

numpy.ndarray that includes:

- Speed to right motor
- Speed to left motor
- Speed to back motor

Example:

```
from robocadSimPy import Funcad


funcad = Funcad.Funcad()
out = funcad.from_axis_to_motors(5, -5, 3)  # [ 2.273672 -9.273672  4.      ]
```

Additional info:

Location and name: "Funcad.h".Funcad.from_axis_to_motors()

Inputs:

- *float* speed to X axis

- *float* speed to Y axis
- *float* speed to Z axis

Output:

*float** that includes:

- Speed to right motor
- Speed to left motor
- Speed to back motor

Example:

```
#include "Funcad.h"
#include <iostream>

int main()
{
    Funcad funcad;
    float* out = funcad.from_axis_to_motors(5, -5, 3); // 2.2735 -9.2735 4
}
```

Additional info:

Location and name: RobocadSim.Funcad.FromAxisToMotors()

Inputs:

- *float* speed to X axis
- *float* speed to Y axis
- *float* speed to Z axis

Output:

System.Numerics.Vector3 that includes:

- Speed to right motor
- Speed to left motor
- Speed to back motor

Example:

```
using System;
using RobocadSim;

namespace TestLib
{
    class Program
    {
        static void Main(string[] args)
        {
            Funcad funcad = new Funcad();
            Vector3 vec = funcad.FromAxisToMotors(5, -5, 3); // 2,273672  -9,273672  4
        }
    }
}
```

Additional info:

4.2 FromMotorsToAxis

FromMotorsToAxis function is used to remake input motors values into axis values for **tricycle** robot.

Python

C++

C#

Location and name: Funcad.Funcad.from_motors_to_axis()

Inputs:

- *float* speed to right motor
- *float* speed to left motor
- *float* speed to back motor

Output:

numpy.ndarray that includes:

- Speed to X axis
- Speed to Y axis
- Speed to Z axis

Example:

```
from robocadSimPy import Funcad


funcad = Funcad.Funcad()
out = funcad.from_motors_to_axis(5, -5, 0)  # [8.66 0.   0.  ]
```

Additional info:

Location and name: "Funcad.h".Funcad.from_motors_to_axis()

Inputs:

- *float* speed to right motor
- *float* speed to left motor
- *float* speed to back motor

Output:

*float** that includes:

- Speed to X axis
- Speed to Y axis
- Speed to Z axis

Example:

```
#include "Funcad.h"
#include <iostream>

int main()
{
    Funcad funcad;
    float* out = funcad.from_motors_to_axis(5, -5, 0); // 8.66 0 0
}
```

Additional info:

Location and name: RobocadSim.Funcad.FromMotorsToAxis()

Inputs:

- *float* speed to right motor
- *float* speed to left motor
- *float* speed to back motor

Output:

System.Numerics.Vector3 that includes:

- Speed to X axis
- Speed to Y axis
- Speed to Z axis

Example:

```
using System;
using RobocadSim;

namespace TestLib
{
    class Program
    {
        static void Main(string[] args)
        {
            Funcad funcad = new Funcad();
            Vector3 vec = funcad.FromMotorsToAxis(5, -5, 0); // 8.66 0 0
        }
    }
}
```

Additional info:

4.3 InRangeBool

InRangeBool function is used to check that input value is in range.

Python

C++

C#

Location and name: Funcad.Funcad.in_range_bool()

Inputs:

- *float* input value
- *float* lower threshold
- *float* upper threshold

Output:

bool is in range

Example:

```python
from robocadSimPy import Funcad


funcad = Funcad.Funcad()
out = funcad.in_range_bool(5, 0, 12)  # True
```

Additional info:

Location and name: "Funcad.h".Funcad.in_range_bool()

Inputs:

- *float* input value
- *float* lower threshold
- *float* upper threshold

Output:

bool is in range

Example:

```cpp
#include "Funcad.h"
#include <iostream>

int main()
{
    Funcad funcad;
    bool out = funcad.in_range_bool(5, 0, 12); // true
}
```

Additional info:

Location and name: RobocadSim.Funcad.InRangeBool()

Inputs:

- *float* input value
- *float* lower threshold
- *float* upper threshold

Output:

bool is in range

Example:

```
using System;
using RobocadSim;

namespace TestLib
{
    class Program
    {
        static void Main(string[] args)
        {
            Funcad funcad = new Funcad();
            bool output = funcad.InRangeBool(5, 0, 12); // true
        }
    }
}
```

Additional info:

4.4 TransfunctionCoda

TransfunctionCoda function is used to rearrange input value. Created by subsystems developer Coda.

Python

C++

C#

Location and name: Funcad.Funcad.transfunc_coda()

Inputs:

- *float* value to remake
- *list* input array
- *list* output array

Output:

float remaded input

Example:

```
from robocadSimPy import Funcad


```

(continues on next page)

(continued from previous page)

```
funcad = Funcad.Funcad()
out = funcad.transfunc_coda(5, [2, 10], [20, 100])  # out will be 50
```

Additional info:

Location and name: "Funcad.h".Funcad.transfunc_coda()

Inputs:

- *float* value to remake
- *float** input array
- *float** output array
- *int* size of input or output array

Output:

float remaded input

Example:

```
#include "Funcad.h"
#include <iostream>

int main()
{
    Funcad funcad;
    float in_arr[] = { 2, 10 };
    float out_arr[] = { 20, 100 };
    float out = funcad.transfunc_coda(5, in_arr, out_arr, 2); // out will be 50
}
```

Additional info:

Location and name: RobocadSim.Funcad.TransfunctionCoda()

Inputs:

- *float* value to remake
- *List<float>* input array
- *List<float>* output array

Output:

float remaded input

Example:

```
using System;
using RobocadSim;

namespace TestLib
{
    class Program
    {
```

(continues on next page)

(continued from previous page)

```
        static void Main(string[] args)
        {
            Funcad funcad = new Funcad();
            float[] inArr = { 2, 10 };
            List<float> inList = new List<float>(inArr);
            float[] outArr = { 20, 100 };
            List<float> outList = new List<float>(outArr);
            float outVal = funcad.TransfunctionCoda(5, inList, outList); // out will
↪be 50
        }
    }
}
```

Additional info:

CHAPTER

FIVE

LIBRARIES DOCUMENTATION

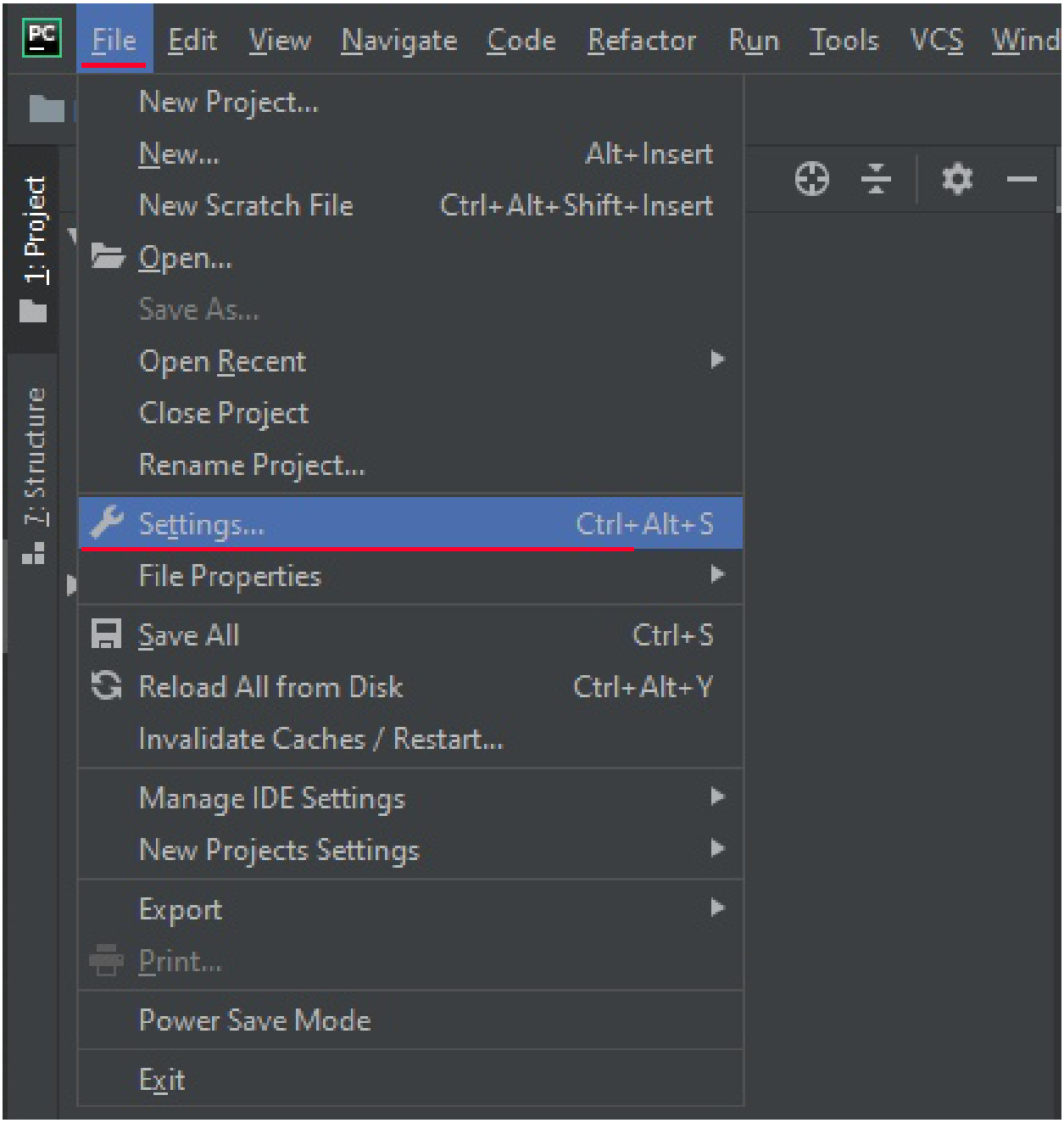
Here You can choose the favourite programming language and read about how to install the library into Your project.

5.1 Python library

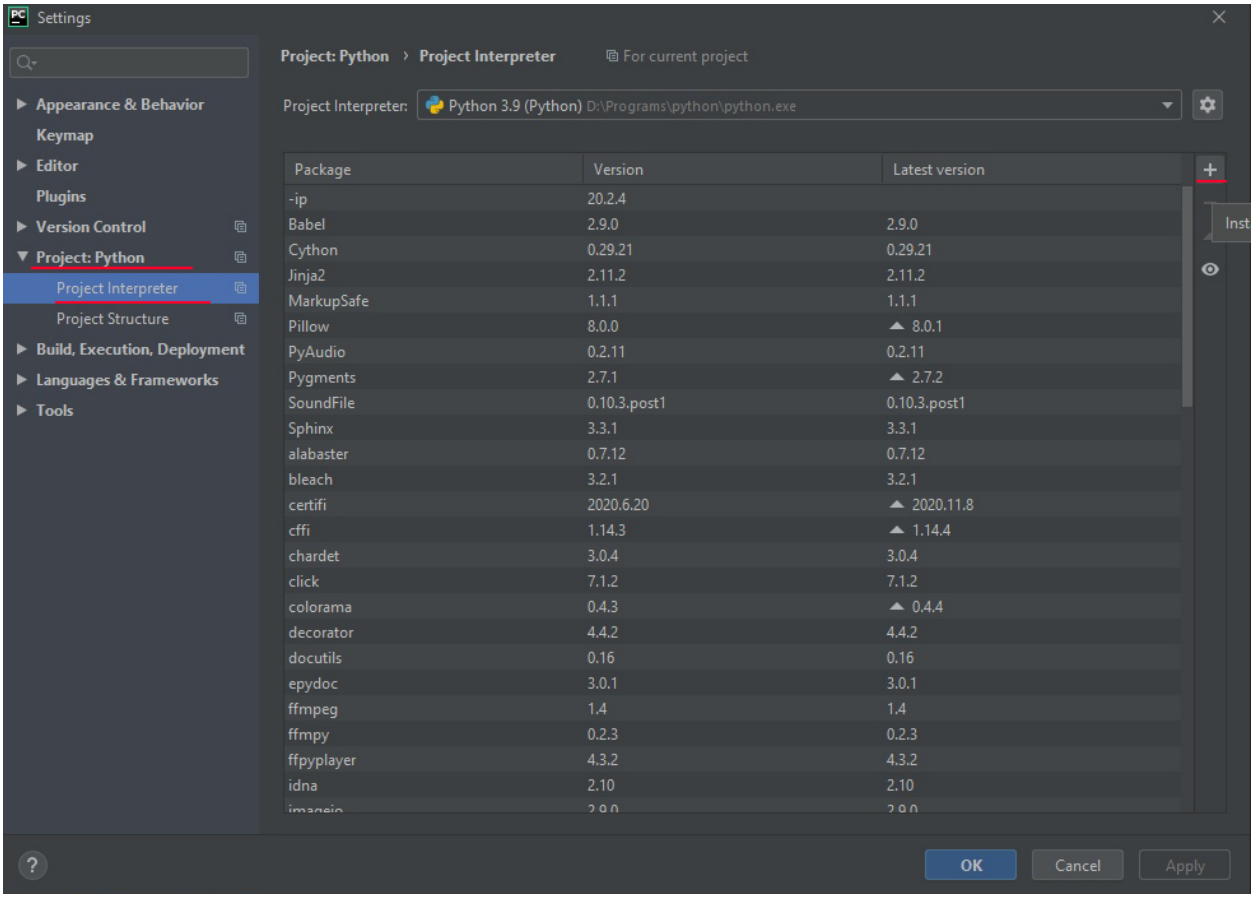
Here is some info about how to download robocadSim Python library. I am going to use PyCharm 2020.

5.1.1 First way:

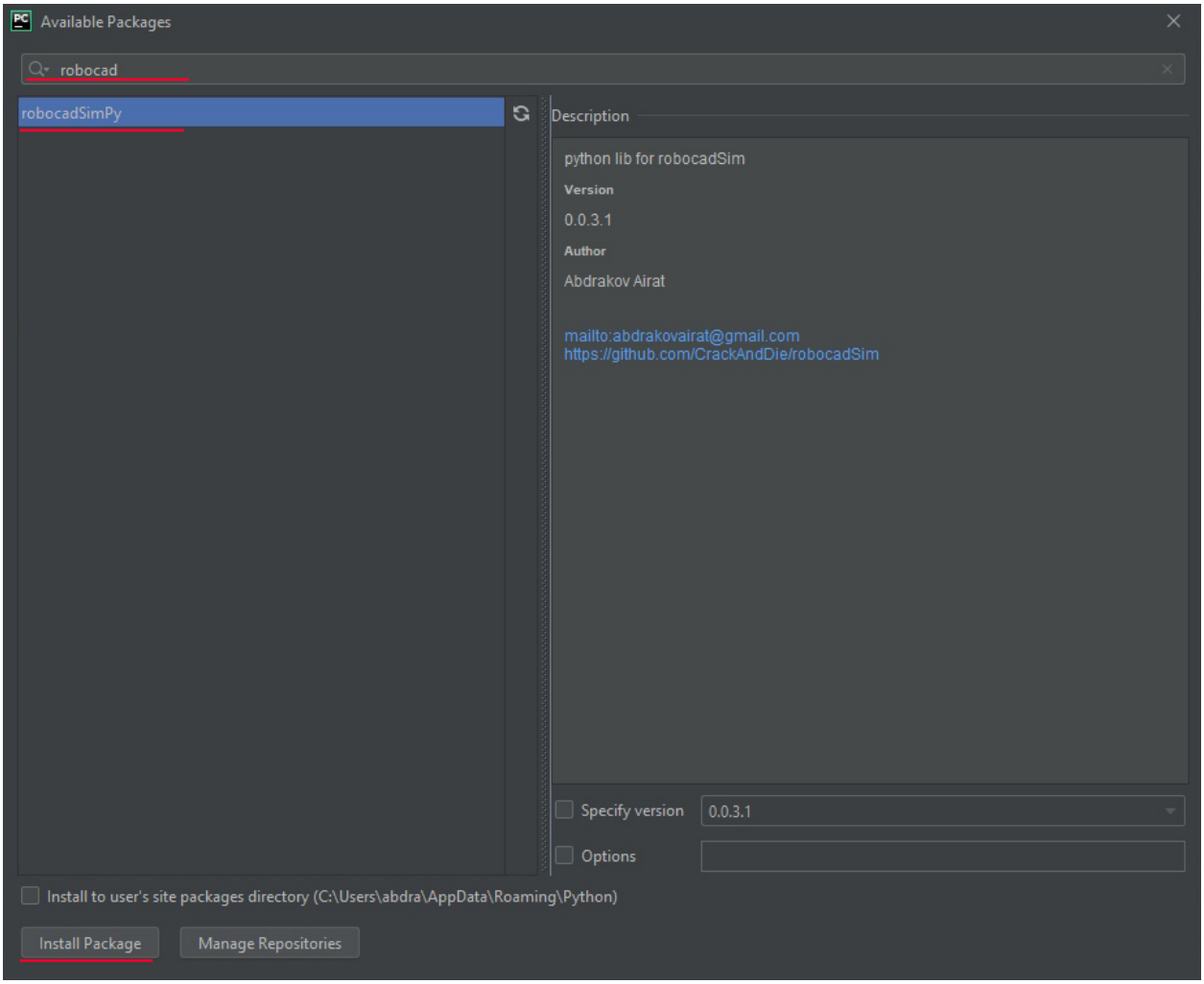
1. Open Your project in PyChram -> click on **File** -> **Settings**



2. Click on **Project: Python** -> **Project Interpreter** -> **Install** (Plus button)



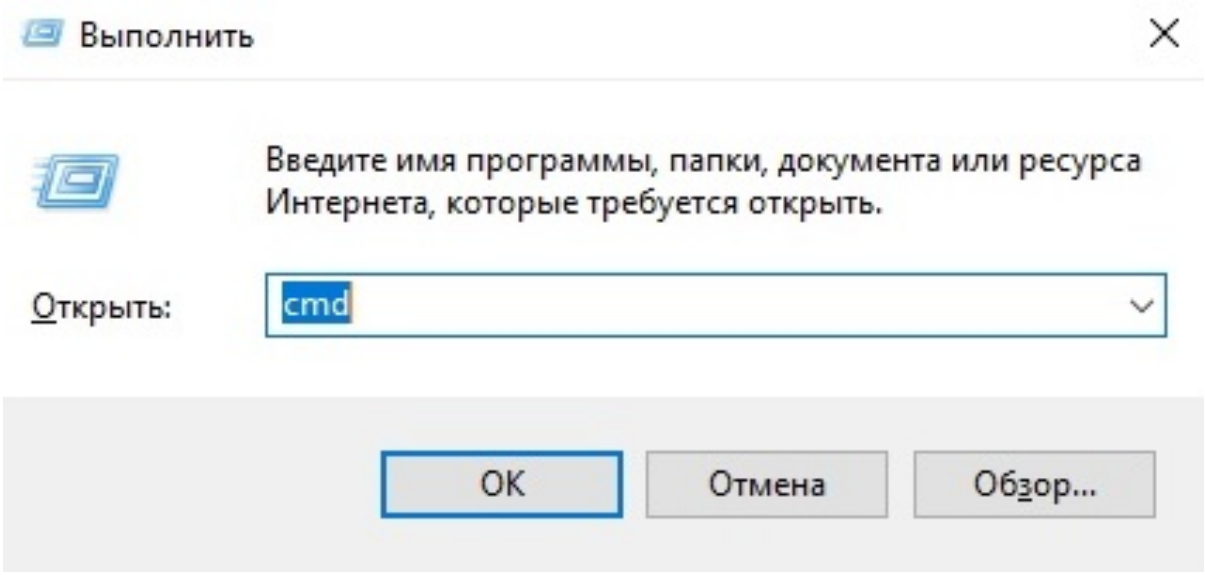
3. Write **robocadSimPy** in **Search Line** -> select **robocadSimPy** -> click **Install Package**



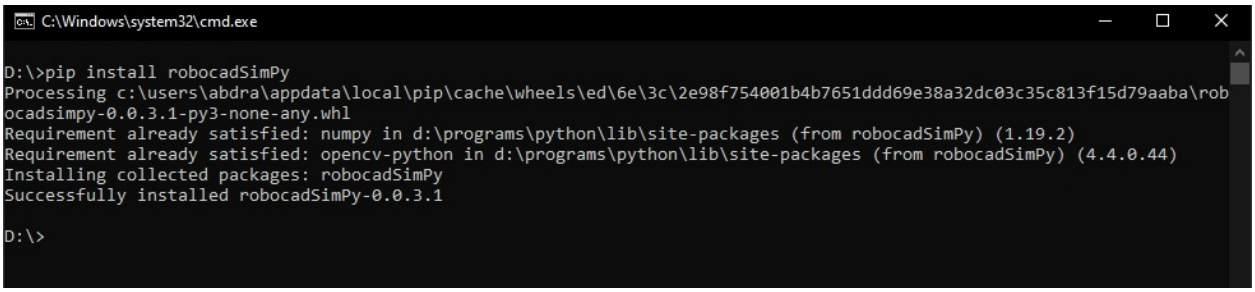
4. Now You can use robocadSim Python library in Your project!

5.1.2 Second way:

1. **Win + R** -> write **cmd** here -> press **Enter**



2. Write here **pip install robocadSimPy** or **pip3 install robocadSimPy** -> press **Enter**



3. Now You can use robocadSim Python library in Your project!

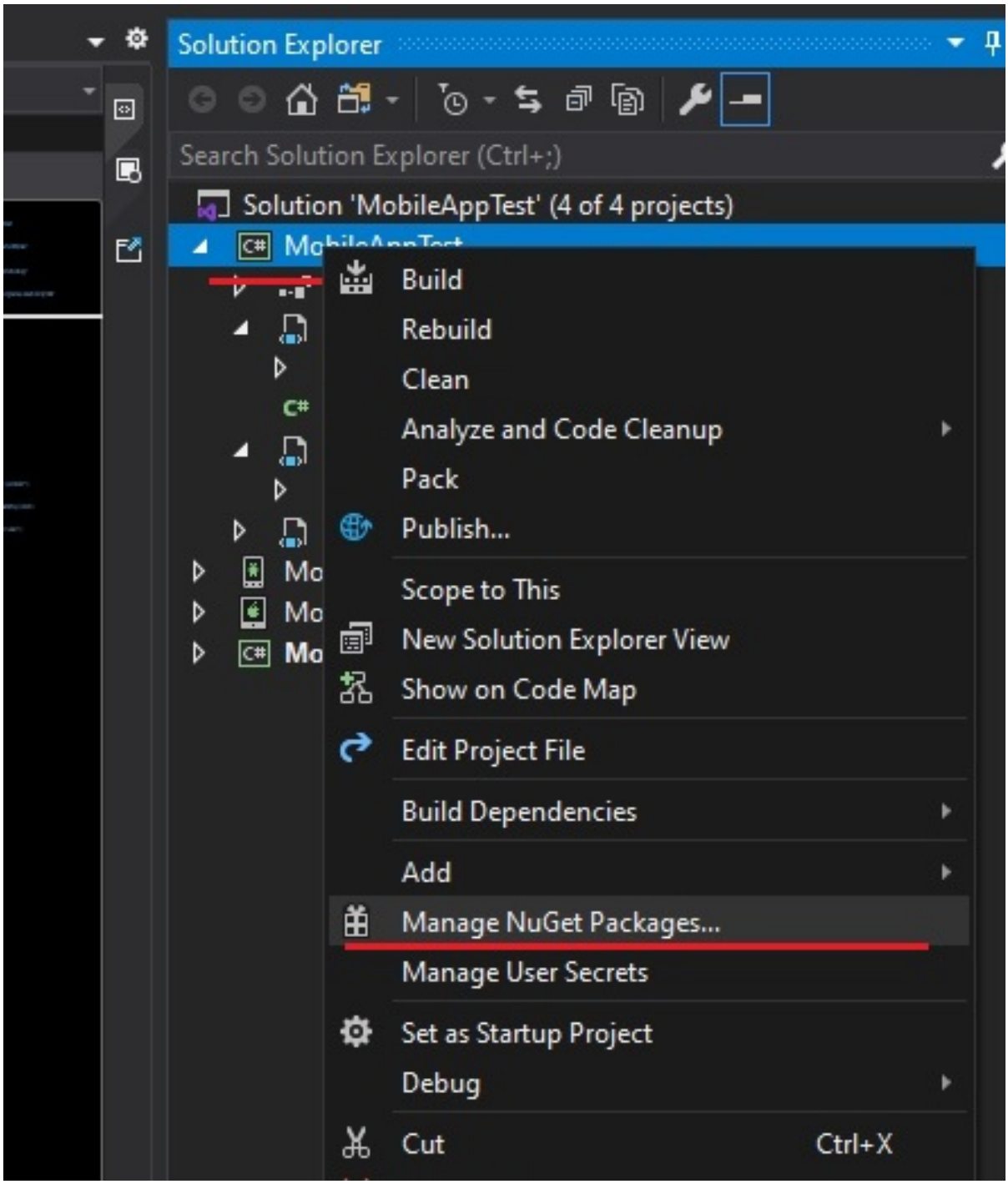
5.2 C# library

Here is some info about how to use robocadSim C# library in your project. I am going to use Visual Studio 2019.

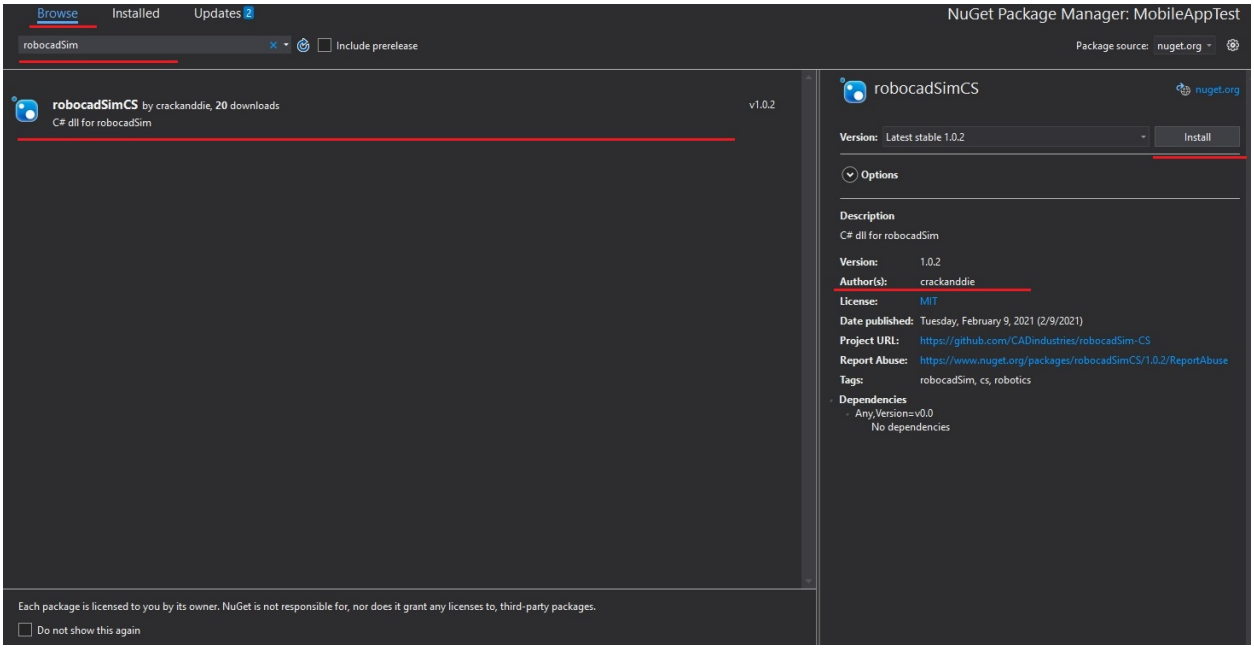
You need emgu-cv installed in your project if You use robocadSim version < v1.3.7. (How to install example).

5.2.1 First way:

1. Right click on Your project name in **Solution explorer** -> click on **Manage NuGet Packages**

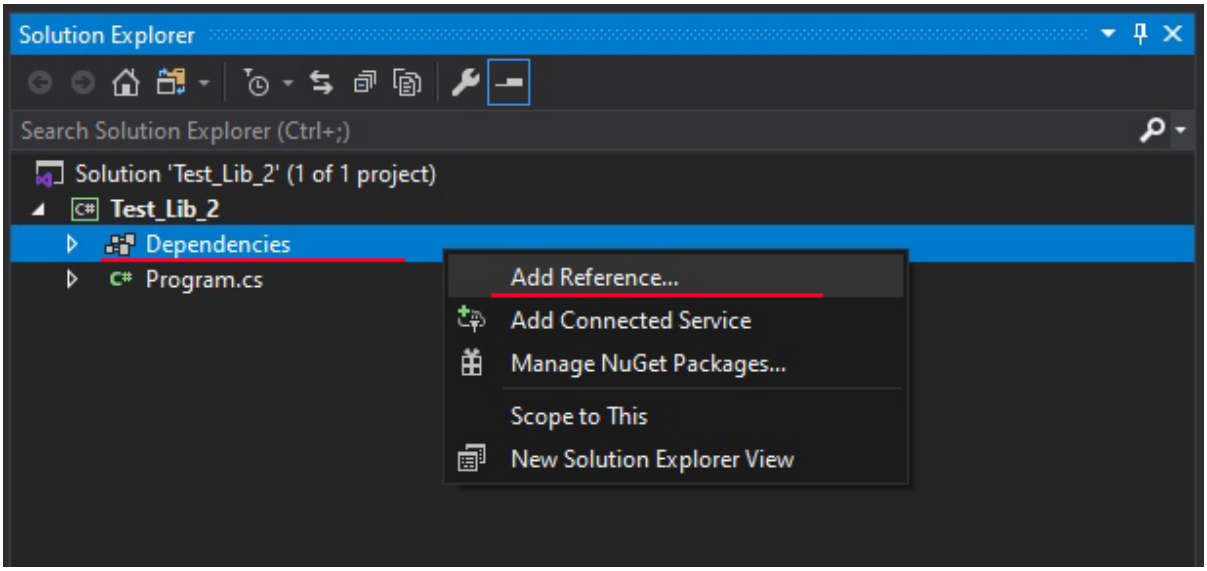


2. Click on **Browse** -> write there **robocadSim** and click Enter -> choose **robocadSimCS** created by **crackanddie** or **Abdrakov corp.** and click on install button

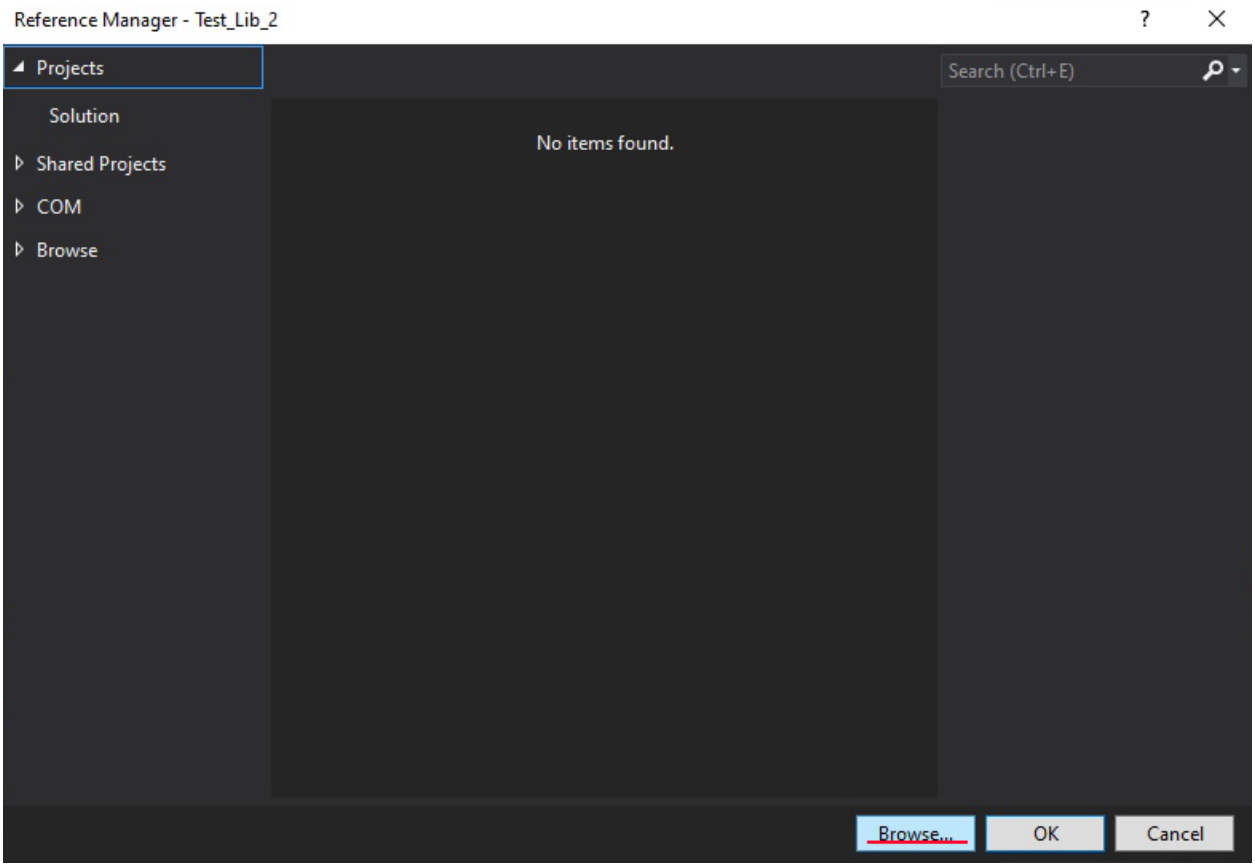


5.2.2 Second way:

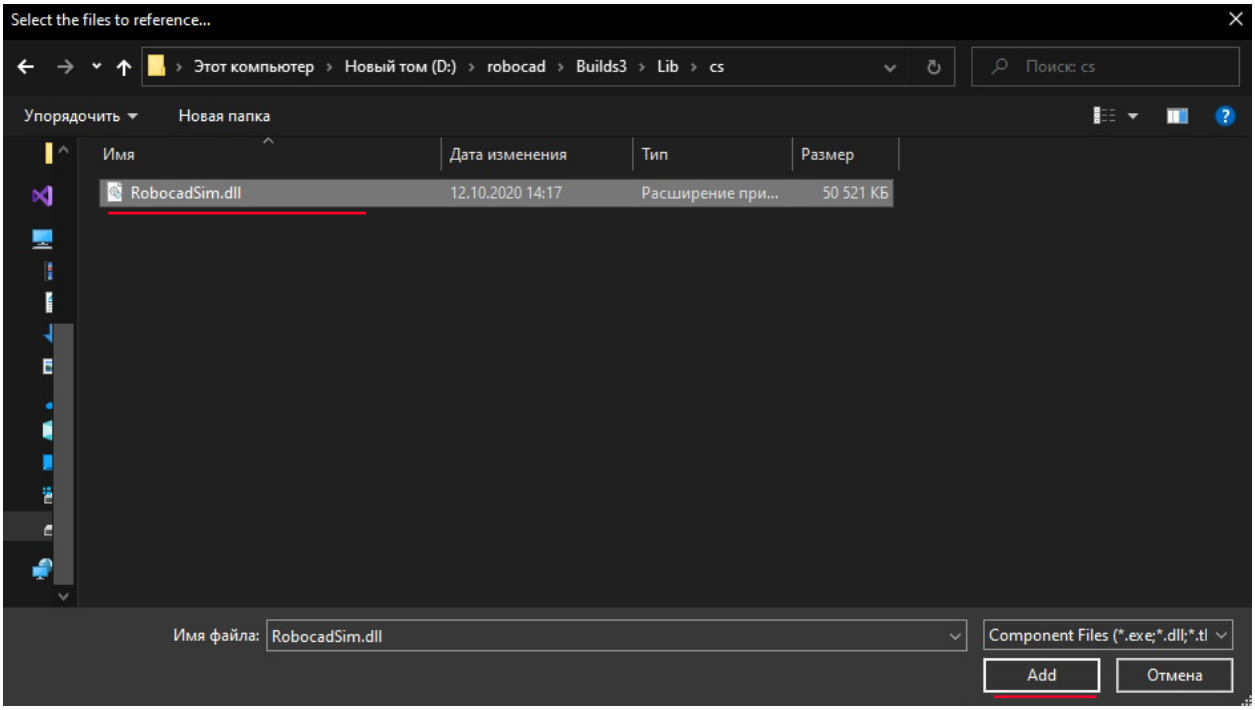
1. Click on Your project name in **Solution explorer** -> right click on **Dependencies** -> **Add reference…**



2. Click on **Browse…**



3. Select **RobocadSim.dll** in **./Lib/cs/** and click **Add**

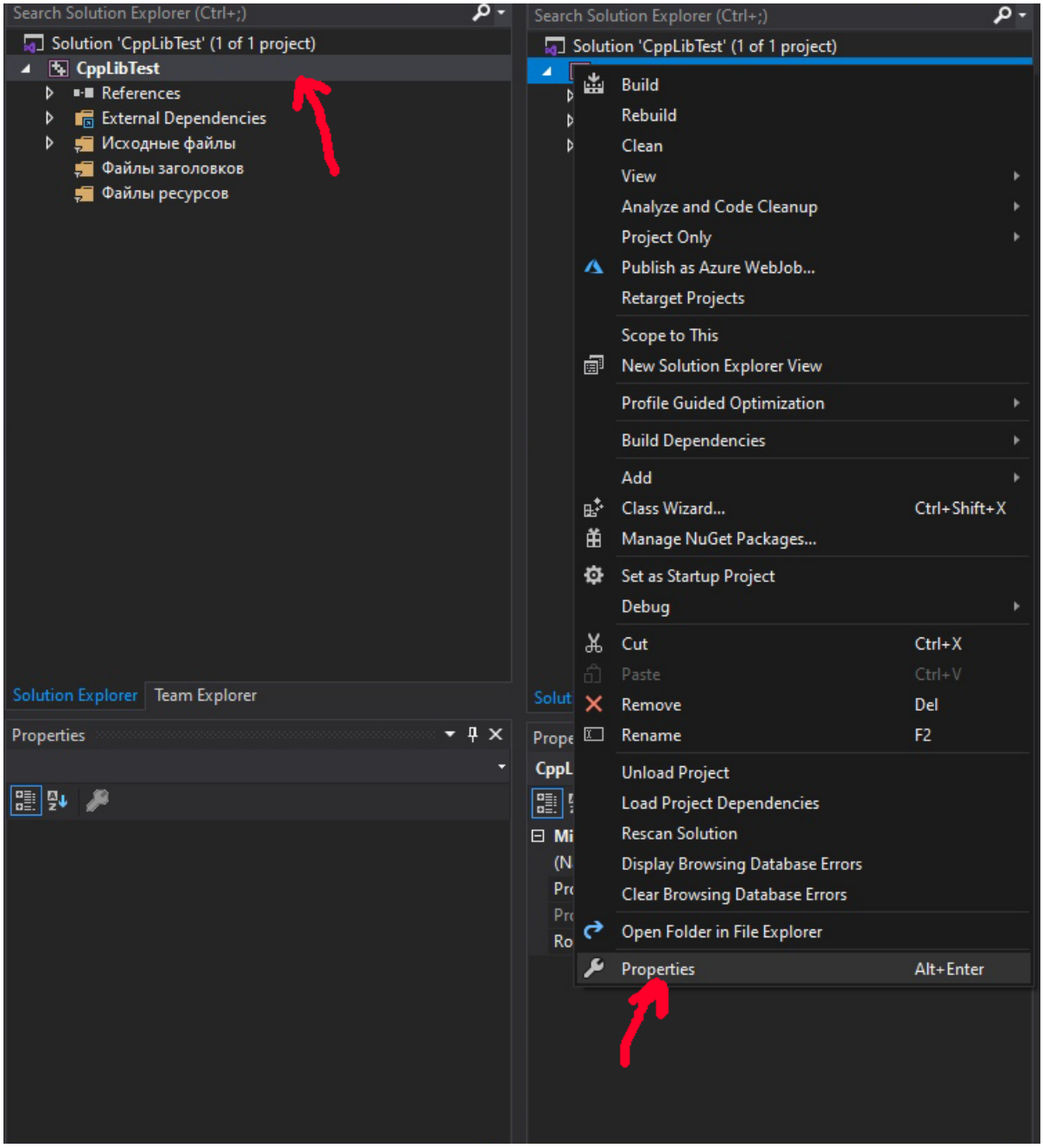


4. Now You can use robocadSim C# library in Your project!

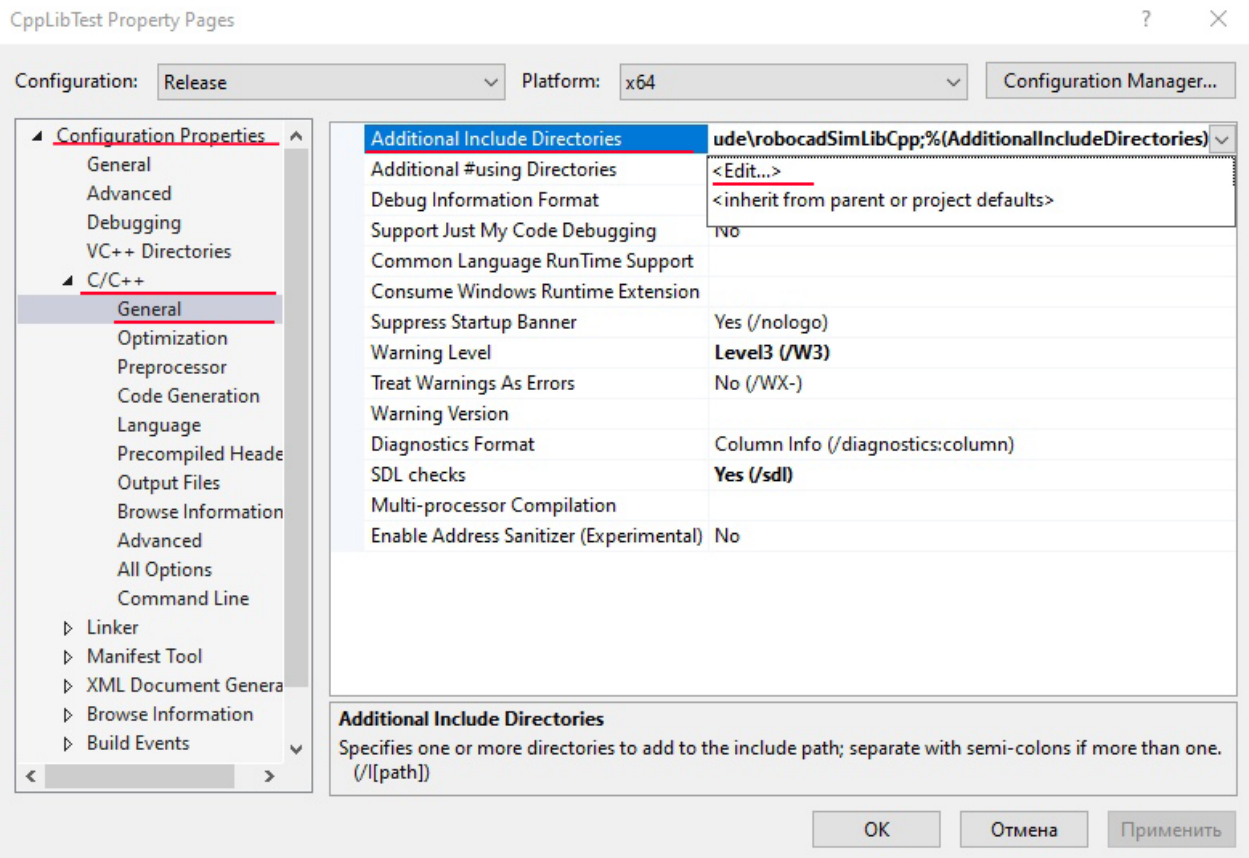
5.3 C++ library

Here is some info about how to use robocadSim C++ library in your project. I am going to use Visual Studio 2019.

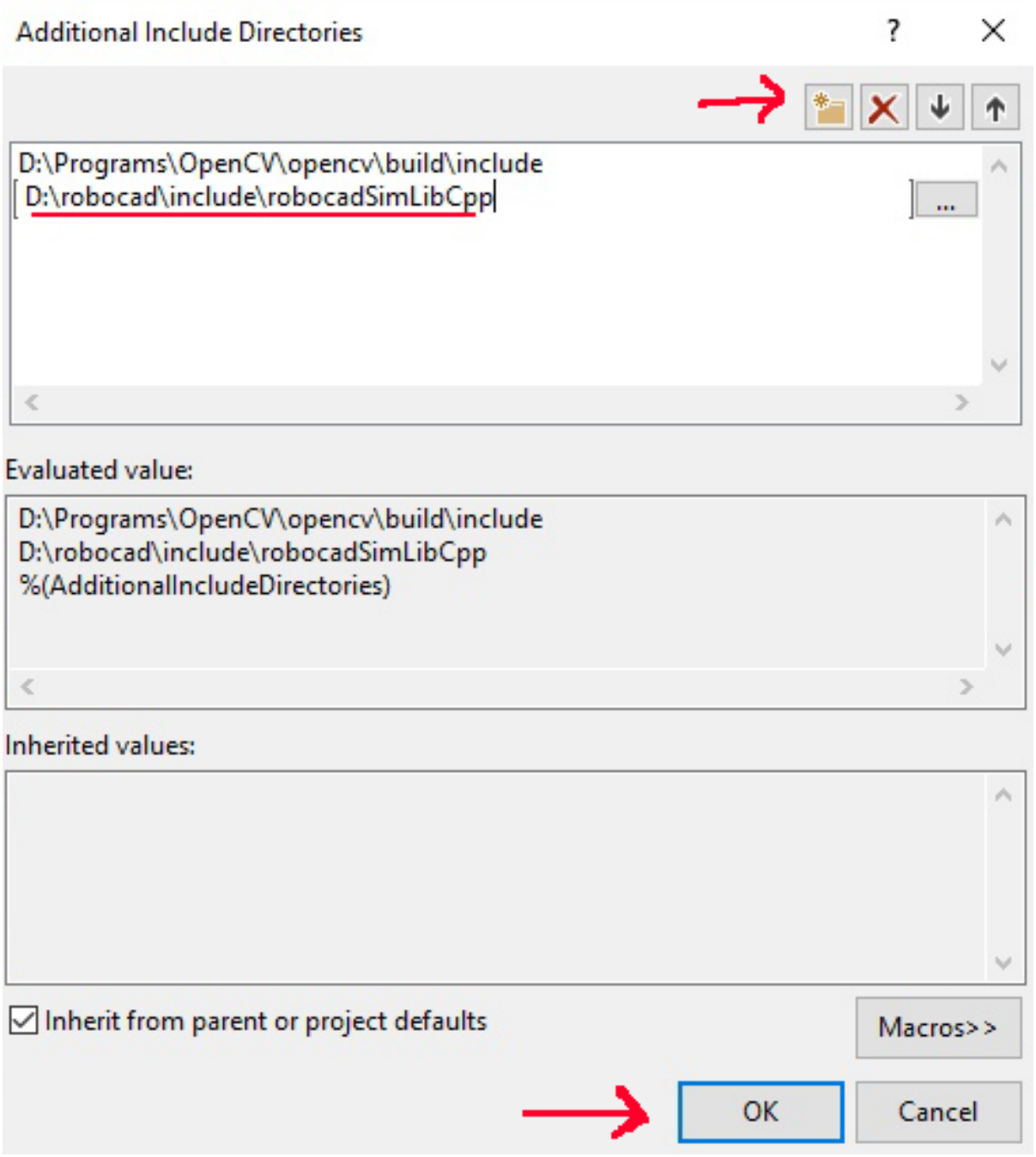
1. You need open-cv installed in your project. (How to install example).
2. Right click on Your project name in **Solution explorer** -> **Properties**



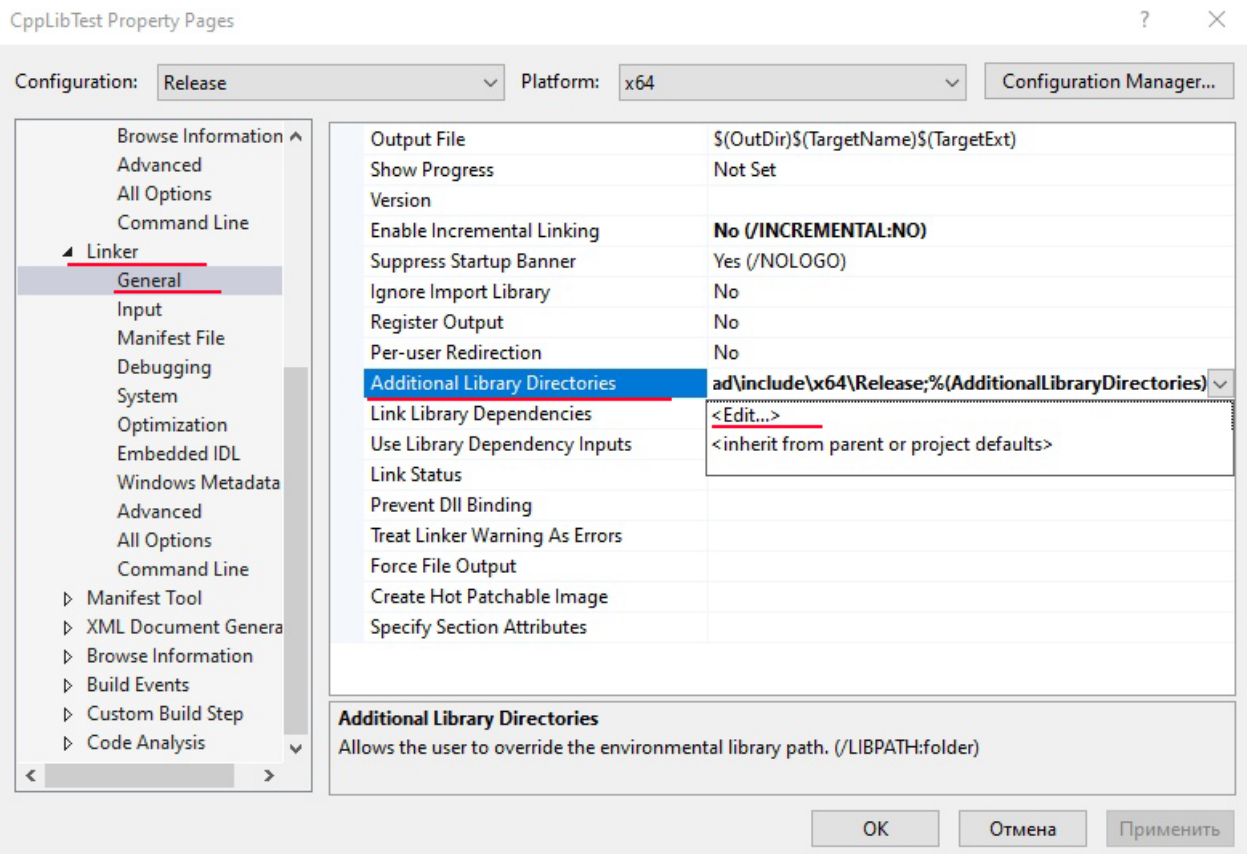
3. Click on **Configuration Properties** -> **C/C++** -> **General** -> **Additional Include Directories** -> **Edit**



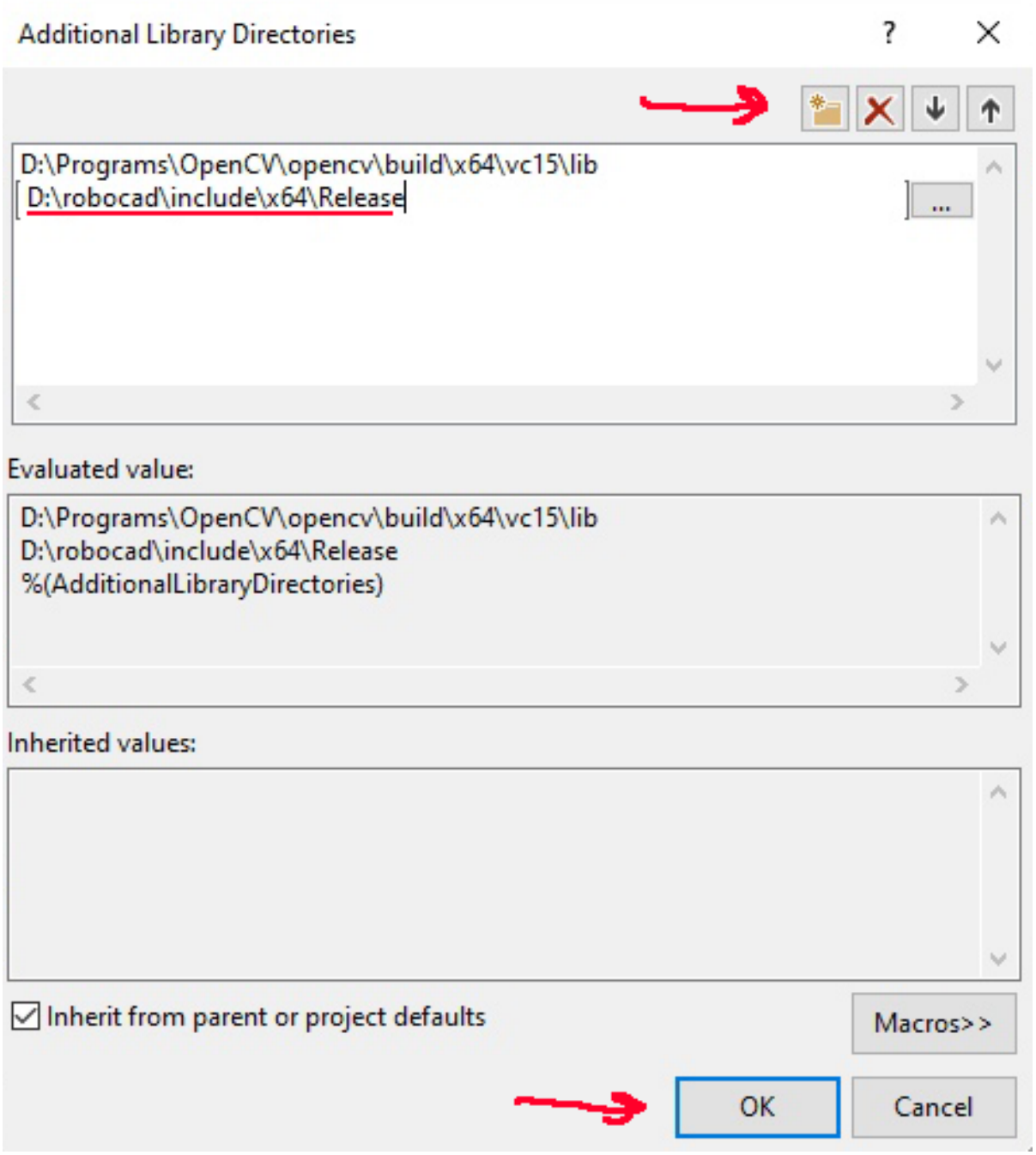
4. Create a new line and paste here path to C++ header files (./robocadSim/Lib/cpp/robocadSimLibCpp) -> click **OK**



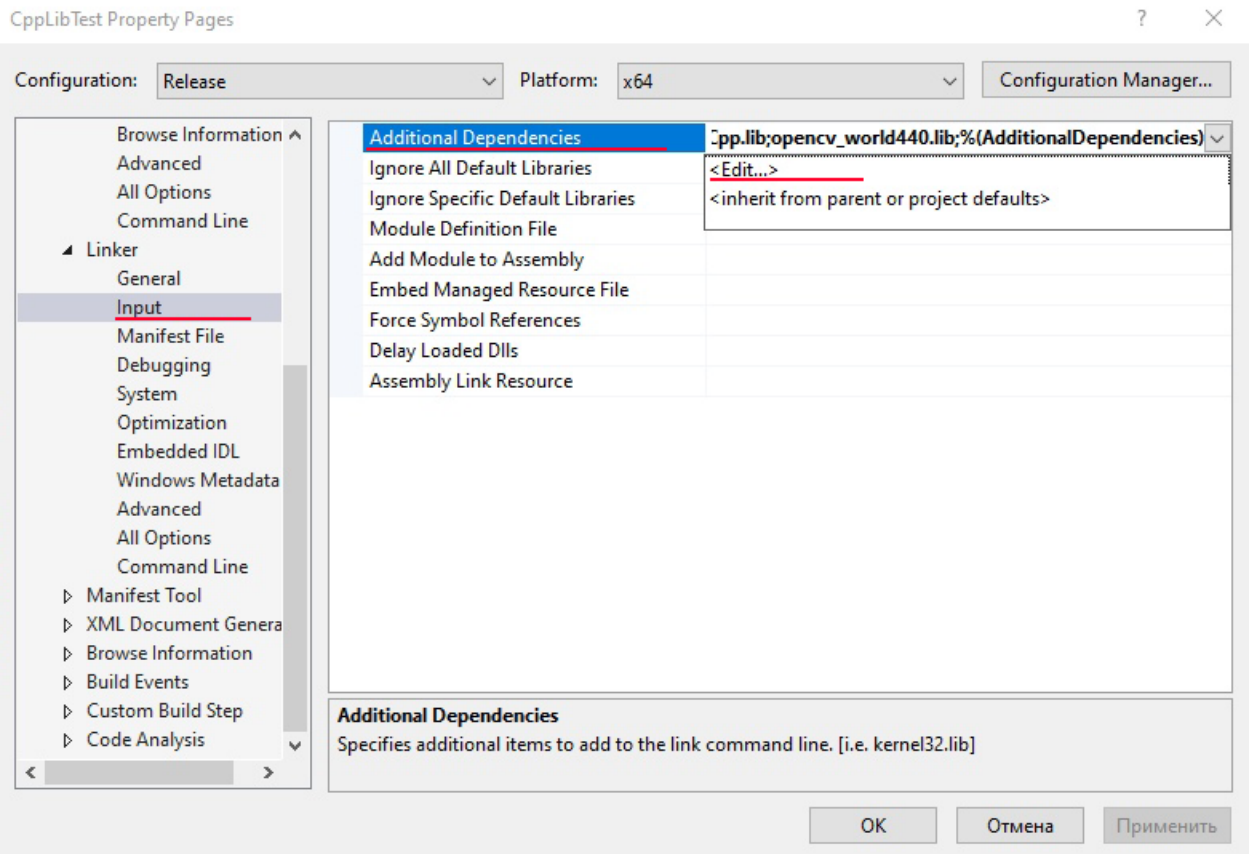
5. Go to **Linker -> General -> Additional Library Directories -> Edit**



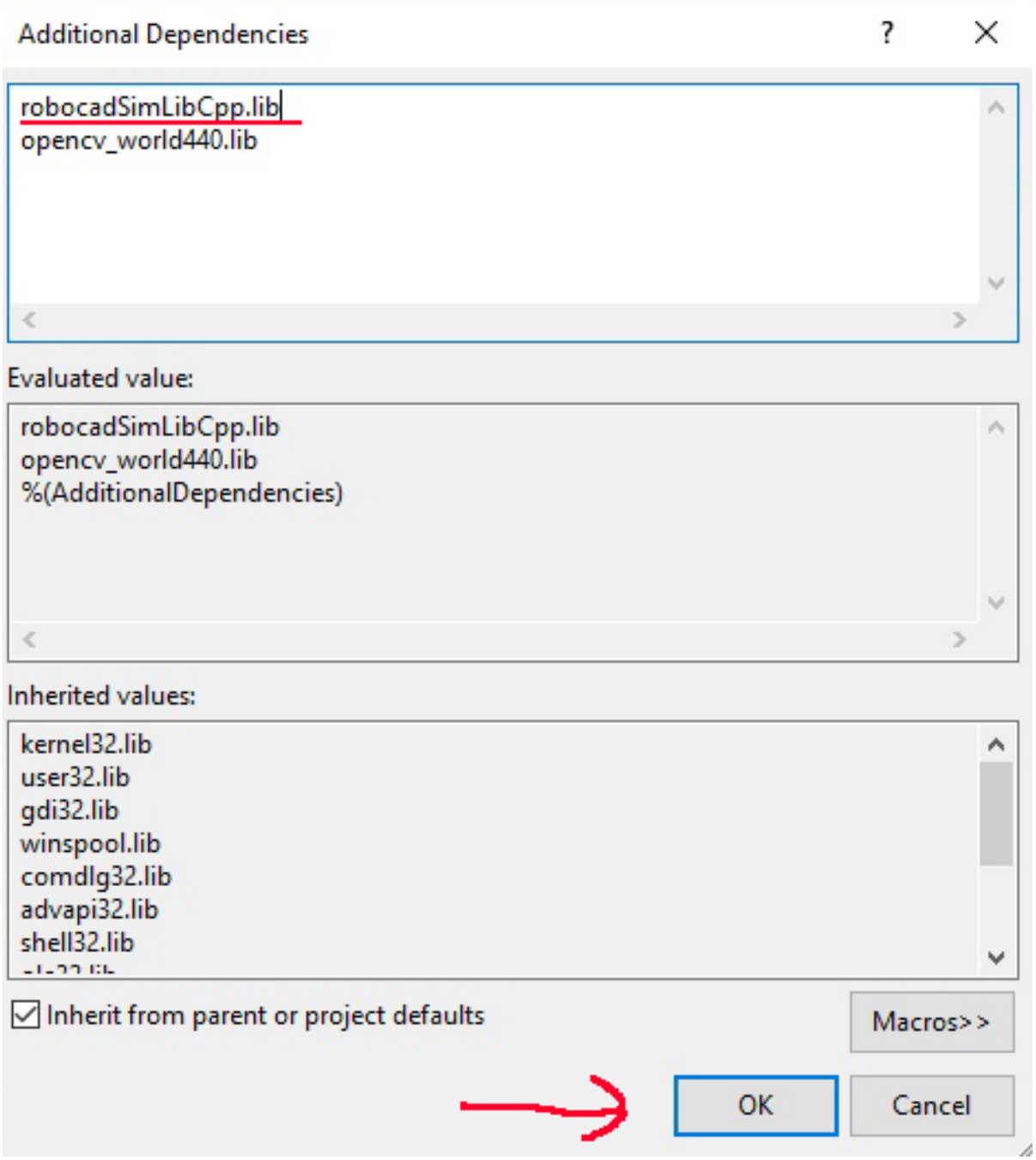
6. Create new line and paste here path to **.lib** file (./robocadSim/Lib/cpp/x64/Release) -> click **OK**



7. Go to **Linker -> Input -> Additional dependencies -> Edit**



8. Paste here robocadSimLibCpp.lib line -> click **OK**



9. Now You can use robocadSim C++ library in Your project!

5.3.1 If You can't use some header files:

1. Copy **.dll** file in robocadSim release folder (./robocadSim/Lib/cpp/x64/Release)
2. Paste it to the path: **path_to_your_project/your_project_name/your_project_name/**

CHAPTER

SIX

ROBOCADSIM DOCUMENTATION

Here You can find some info about robocadSim program.

6.1 Hot Keys and Keys

6.1.1 Keys:

- **Esc:** open “pause” menu

6.1.2 Hot Keys:

- **LeftCtrl + E:** open power panel
- **LeftCtrl + D:** open control panel

CHAPTER

SEVEN

BE OUR SPONSOR

If you want to help the project financially, then you can send money to any of these accounts:

- **PayPal:** paypal.me/crackanddie
- **Qiwi:** qiwi.com/p/79656098785
- **Visa:** 4000 7934 7377 7474

Thanks!

CHAPTER

EIGHT

NEED HELP

If You need for help please contact me or open an issue on GitHub:

- Inst: robocadSim
- Email: robocadsim@gmail.com
- Facebook: RobocadSim

CHAPTER

NINE

LICENSE

MIT License

Copyright (c) 2020 Airat

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.