

LABOR JEGYZŐKÖNYV

MOBIL ROBOT ÉPÍTÉSE

Labor helye: Széchenyi István Egyetem, L4--3 laboratórium,
9026 Győr, Egyetem tér 1.
Labor ideje: 2021. 10. 18.
Labor tárgya: Robot építése, programozása
Labort végezte: Füleki Tamás, Császár Miksa Henrik, Balbach Dominik
Labor megrendelője: Hajdu Csaba

Feladat meghatározása:

1. Egy saját ROS2 package létrehozása, amelyben legalább kettő Python-alapú node legyen, az egyik node szabadon választott szabályozást valósít meg, míg a többi csomópont a robot szenzor-aktuátor interfészét valósítja meg.
 2. Webots nevű szimulátorban kell egy robotot létrehozni, ami legalább két kerekű.

Kivitelezés:

1.Feladat:

A ROS2 package két node-ból áll, az egyiket robot-nak, a másikat pid-nek hívják.

A robot node az előzőekben megépített LEGO robotunkat szimulálja.

A robot 11 tonikot tartalmaz, melyek a különböző szenzorokat, illetve motorokat reprezentálják. A node ezeknek a szenzoroknak megfelelően generál véletlenszerű adatokat, melyeket 0.5 másodpercenként továbbítunk a pid node-nak, illetve kiírjuk az elküldött értéküket.

Link a robot.py-hez

Segéd függvények

```

47
48     def genSin(res, j=1): # impact
49         x = np.linspace(-np.pi, np.pi, res)
50         y = np.sin(x*j)
51         return (x, y)
52
53     def getRandInRange(min, step, max):
54         return random.randrange(start=min, step=step, stop=max)
55
56     def myMap(value, leftMin, leftMax, rightMin, rightMax):
57         leftSpan = leftMax - leftMin
58         rightSpan = rightMax - rightMin
59         valueScaled = float(value - leftMin) / float(leftSpan)
60         return rightMin + (valueScaled * rightSpan)

```

Tonikok inicializálása

```

62     class Robot(Node):
63
64         def __init__(self):
65             super().__init__('robot')
66             # initialize topics
67             self.topic_01 = self.create_publisher(Float64, 'get_amotorspeed', 10)
68             self.topic_02 = self.create_publisher(Float64, 'get_amotordeg', 10)
69             self.topic_03 = self.create_publisher(Float64, 'get_bmotorspeed', 10)
70             self.topic_04 = self.create_publisher(Float64, 'get_bmotordeg', 10)
71             self.topic_05 = self.create_publisher(Float64, 'get_cmotorspeed', 10)
72             self.topic_06 = self.create_publisher(Float64, 'get_cmotordeg', 10)
73             self.topic_07 = self.create_publisher(Int32, 'get_ultrasonicraw', 10)
74             self.topic_08 = self.create_publisher(Bool, 'get_touchsensor', 10)
75             self.topic_09 = self.create_publisher(Int32, 'get_rgbsensor_r', 10)
76             self.topic_10 = self.create_publisher(Int32, 'get_rgbsensor_g', 10)
77             self.topic_11 = self.create_publisher(Int32, 'get_rgbsensor_b', 10)
78
79             # initialize subscribers
80             self.sub_01 = self.create_subscription(Int8,\n81                 'set_direction', self.listen_set_direction, 10)
82             self.sub_02 = self.create_subscription(Float64,\n83                 'set_speed', self.listen_set_speed, 10)
84             self.sub_03 = self.create_subscription(Int32,\n85                 'set_ul_real', self.listen_set_ul, 10)
86
87             # self.publisher_ul = self.create_publisher(Float64, 'ul', 10)
88             # timer sends packets in every 0.5 seconds
89             timer_period = 0.5 # seconds
90             self.timer = self.create_timer(timer_period, self.timer_callback)
91             self.computeUp = self.create_timer(0.1, self.computeUp_cr)
92             self.i = 0

```

Helyi változók inicializálása, véletlen adatok generálása.

```
89         timer_period = 0.5 # seconds
90         self.timer = self.create_timer(timer_period, self.timer_callback)
91         self.computeUp = self.create_timer(0.1, self.computeUp_cr)
92         self.i = 0
93
94         # Init local variables
95
96         self._as = 0
97         self._ad = 0
98         self._bs = 0
99         self._bd = 0
100        self._cs = 0
101        self._cd = 0
102        self._ul = 0
103        self._ts = 0
104        self._cr = 0
105        self._cg = 0
106        self._cb = 0
107
108        # Init received variables
109
110        self._l_as = 0
111        self._l_bs = 0
112        self._l_ul = 0
113
114        self._dir = 0
115        self._dir_speed = 0
116
117    def genDummyData(self):
118        self._as = float(random.randrange(start=-100, stop=100, step=1))
119        self._ad = float(random.randrange(start=0, stop=360, step=1))
120        self._bs = float(random.randrange(start=-100, stop=100, step=1))
121        self._bd = float(random.randrange(start=0, stop=360, step=1))
122        self._cs = float(random.randrange(start=-100, stop=100, step=1))
123        self._cd = float(random.randrange(start=-180, stop=180, step=1))
124        self._ul = random.randrange(start=0, stop=4096, step=1)
125        self._ts = bool(random.getrandbits(1))
126        self._cr = random.randrange(start=0, stop=4096, step=1)
127        self._cg = random.randrange(start=0, stop=4096, step=1)
128        self._cb = random.randrange(start=0, stop=4096, step=1)
```

A visszatérő adatok befogadása.

```

117     def genDummyData(self):
118         self._as = float(random.randrange(start=-100, stop=100, step=1))
119         self._ad = float(random.randrange(start=0, stop=360, step=1))
120         self._bs = float(random.randrange(start=-100, stop=100, step=1))
121         self._bd = float(random.randrange(start=0, stop=360, step=1))
122         self._cs = float(random.randrange(start=-100, stop=100, step=1))
123         self._cd = float(random.randrange(start=-180, stop=180, step=1))
124         self._ul = random.randrange(start=0, stop=4096, step=1)
125         self._ts = bool(random.getrandbits(1))
126         self._cr = random.randrange(start=0, stop=4096, step=1)
127         self._cg = random.randrange(start=0, stop=4096, step=1)
128         self._cb = random.randrange(start=0, stop=4096, step=1)
129
130     def listen_set_direction(self, msg):
131         self._dir = msg.data
132     def listen_set_speed(self, msg):
133         self._dir_speed = msg.data
134     def listen_set_ul(self, msg):
135         self._l_ul = msg.data

```

Miután a pid node feldolgozta az adatokat, visszaküldi a robot node-nak és az hasznosítja őket oly formában, hogy a robot az előzőekben megfelelően működjön.

```

137     def computeUp_cr(self):
138         if self._dir == 1:
139             self._l_as = 50
140             self._l_bs = self._cd
141             if self._l_ul < 25:
142                 self._l_as = self._dir_speed
143                 self._l_bs = self._dir_speed
144             elif self._dir == -1:
145                 self._l_as = -50
146                 self._l_bs = self._cd
147                 if self._l_ul < 25:
148                     self._l_as = self._dir_speed
149                     self._l_bs = self._dir_speed
150             elif self._dir == 0:
151                 self._l_as = -20
152                 self._l_bs = -20
153             else:
154                 self._l_as = self._dir_speed
155                 self._l_as = self._dir_speed
156             self.get_logger().info(
157                 f'The robot go {"Left" if self._dir == 1 else "Right" if self._dir == -1 else "Straight"}' \\
158                 +f' at {(self._l_as, self._l_bs)} speed'
159         )

```

Az időzített küldő funkció

```
162     def timer_callback(self):
163         self.genDummyData()
164         # Messages types
165         msg_amotorspeed = Float64()
166         msg_amotordeg = Float64()
167         msg_bmotorspeed = Float64()
168         msg_bmotordeg = Float64()
169         msg_cmotorspeed = Float64()
170         msg_cmotordeg = Float64()
171
172         msg_ultrasonicraw = Int32()
173
174         msg_touchsensor = Bool()
175
176         msg_rgbsensor_r = Int32()
177         msg_rgbsensor_g = Int32()
178         msg_rgbsensor_b = Int32()
179
180         #msg.data = 'Hello World: %d' % self.i
181         #msg.data = self.robot.msgData() + str(self.i)
182         # Get sensors data
183         msg_amotorspeed.data = self._as
184         msg_amotordeg.data = self._ad
185         msg_bmotorspeed.data = self._bs
186         msg_bmotordeg.data = self._bd
187         msg_cmotorspeed.data = self._cs
188         msg_cmotordeg.data = self._cd
```

Az időzített küldő folytatása, adatok kiírása.

```
199     msg_ultrasonicraw.data = self._ul
200     msg_touchsensor.data = self._ts
201     #r = int(random.randrange(0, 4096, 1))
202     #g = int(random.randrange(0, 4096, 1))
203     #b = int(random.randrange(0, 4096, 1))
204     #r = 4096
205     #g = 4096
206     #b = 4096
207
208     #msg_rgbsensor.data = (r<<24|g<<12|b)
209     msg_rgbsensor_r.data = self._cr
210     msg_rgbsensor_g.data = self._cg
211     msg_rgbsensor_b.data = self._cb
212
213     #msg_motor.data = float(random.random())
214     #self.robot.Tick(self.i)
215
216     self.topic_01.publish(msg_amotorspeed)
217     self.topic_02.publish(msg_amotordeg)
218     self.topic_03.publish(msg_bmotorspeed)
219     self.topic_04.publish(msg_bmotordeg)
220     self.topic_05.publish(msg_cmotorspeed)
221     self.topic_06.publish(msg_cmotordeg)
222     self.topic_07.publish(msg_ultrasonicraw)
223     self.topic_08.publish(msg_touchsensor)
224     self.topic_09.publish(msg_rgbsensor_r)
225     self.topic_10.publish(msg_rgbsensor_g)
226     self.topic_11.publish(msg_rgbsensor_b)
227     #self.publisher_ul.publish(msg_ul)
228     #self.get_logger().info('-----')
229     self.get_logger().info(f'[{self.i}]Publishing topic_01: "{msg_amotorspeed.data}"')
230     self.get_logger().info(f'[{self.i}]Publishing topic_02: "{msg_amotordeg.data}"')
231     self.get_logger().info(f'[{self.i}]Publishing topic_03: "{msg_bmotorspeed.data}"')
232     self.get_logger().info(f'[{self.i}]Publishing topic_04: "{msg_bmotordeg.data}"')
233     self.get_logger().info(f'[{self.i}]Publishing topic_05: "{msg_cmotorspeed.data}"')
234     self.get_logger().info(f'[{self.i}]Publishing topic_06: "{msg_cmotordeg.data}"')
235     self.get_logger().info(f'[{self.i}]Publishing topic_07: "{msg_ultrasonicraw.data}"')
236     self.get_logger().info(f'[{self.i}]Publishing topic_08: "{msg_touchsensor.data}"')
237     self.get_logger().info(f'[{self.i}]Publishing topic_09: "{msg_rgbsensor_r.data}"')
238     self.get_logger().info(f'[{self.i}]Publishing topic_10: "{msg_rgbsensor_g.data}"')
239     self.get_logger().info(f'[{self.i}]Publishing topic_11: "{msg_rgbsensor_b.data}"')
```

A pid node feliratkozik, az első node 11 tonikjára amiket feldolgoz, majd visszaküld a robot node-nak.

<code></code >

```
246 def main(args=None):
247     rclpy.init(args=args)
248
249     robot = Robot()
250
251     rclpy.spin(robot)
252
253     # Destroy the node explicitly
254     # (optional - otherwise it will be done automatically
255     # when the garbage collector destroys the node object)
256     robot.destroy_node()
257     rclpy.shutdown()
258
259
260 if __name__ == '__main__':
261     main()
```

Az adatok feldolgozása és kiíratása egy olyan függvényben történik, mely bekéri a változókat, az ezekkel a változókkal végrehajtani kívánt transzformációt, a loggolás érdekében a változók neveit, illetve ha vissza szeretnénk az eredményt küldeni az előző node-nak, akkor a visszaküldő függvényt a topic változóját, illetve az üzenet típusát is meg kell adnunk.

Az adatokat 0.2 másodpercenként küldjük vissza a robot node-nak.

Könyvtárak bekérése, segédfüggvény.

Link a pid.py-hez

A feliratkozók inicializálása.

```

50   class Pid(Node):
51
52     def __init__(self):
53       super().__init__('pid')
54
55     #subscription = node.create_subscription(
56     #String, 'topic', lambda msg: node.get_logger().info('I heard: "%s"' % msg.data), 10)
57
58     # Init subscribers
59
60     self.sub_01 = self.create_subscription(Float64, 'get_amotorspeed', self.listener_callback_as, 10)
61     self.sub_02 = self.create_subscription(Float64, 'get_amotordeg', self.listener_callback_ad, 10)
62     self.sub_03 = self.create_subscription(Float64, 'get_bmotorspeed', self.listener_callback_bs, 10)
63     self.sub_04 = self.create_subscription(Float64, 'get_bmotordeg', self.listener_callback_bd, 10)
64     self.sub_05 = self.create_subscription(Float64, 'get_cmotorspeed', self.listener_callback_cs, 10)
65     self.sub_06 = self.create_subscription(Float64, 'get_cmotordeg', self.listener_callback_cd, 10)
66     self.sub_07 = self.create_subscription(Int32, 'get_ultrasonicraw', self.listener_callback_ul, 10)
67     self.sub_08 = self.create_subscription(Bool, 'get_touchsensor', self.listener_callback_ts, 10)
68     self.sub_09 = self.create_subscription(Int32, 'get_rgbsensor_r', self.listener_callback_cr, 10)
69     self.sub_10 = self.create_subscription(Int32, 'get_rgbsensor_g', self.listener_callback_cg, 10)
70     self.sub_11 = self.create_subscription(Int32, 'get_rgbsensor_b', self.listener_callback_cb, 10)
71
72     # Init publishers
73
74     self.pub_01 = self.create_publisher(Int8, 'set_direction', 10)
75     self.pub_02 = self.create_publisher(Float64, 'set_speed', 10)
76     self.pub_03 = self.create_publisher(Int32, 'set_ul_real', 10)
77
78     self.timer = self.create_timer(0.2, self.compute_data)
79
80     #self.sub_01 # prevent unused variable warning

```

Helyi változók inicializálása

82	<code>self._as = [0]</code>
83	<code>self._ad = [0]</code>
84	<code>self._bs = [0]</code>
85	<code>self._bd = [0]</code>
86	<code>self._cs = [0]</code>
87	<code>self._cd = [0]</code>
88	<code>self._ul = [0]</code>
89	<code>self._ts = [0]</code>
90	<code>self._cr = [0]</code>
91	<code>self._cg = [0]</code>
92	<code>self._cb = [0]</code>
93	<code>#plt.show(block=True)</code>

A beérkezett változók eltárolása, illetve azok utolsó 10 értékének.

```
94
95     def listener_callback_as(self, msg):
96         self._as.append(msg.data)
97         self._as = self._as[-9:]
98     def listener_callback_ad(self, msg):
99         self._ad.append(msg.data)
100        self._ad = self._ad[-9:]
101    def listener_callback_bs(self, msg):
102        self._bs.append(msg.data)
103        self._bs = self._bs[-9:]
104    def listener_callback_bd(self, msg):
105        self._bd.append(msg.data)
106        self._bd = self._bd[-9:]
107    def listener_callback_cs(self, msg):
108        self._cs.append(msg.data)
109        self._bd = self._bd[-9:]
110    def listener_callback_cd(self, msg):
111        self._cd.append(msg.data)
112        self._cd = self._cd[-9:]
113    def listener_callback_ul(self, msg):
114        self._ul.append(msg.data)
115        self._ul = self._ul[-9:]
116    def listener_callback_ts(self, msg):
117        self._ts.append(msg.data)
118        self._ts = self._ts[-9:]
119    def listener_callback_cr(self, msg):
120        self._cr.append(msg.data)
121        self._cr = self._cr[-9:]
122    def listener_callback_cg(self, msg):
123        self._cg.append(msg.data)
124        self._cg = self._cg[-9:]
125    def listener_callback_cb(self, msg):
126        self._cb.append(msg.data)
127        self._cb = self._cb[-9:]
128
129    def sendPub(self, where, type, msg):
130        obj = type()
131        obj.data = msg
132        #method = getattr(Pid.__init__, where)
133        #print(type(where))
134        where.publish(obj)
```

A compute logger funkció, melyben loggoljuk, és végrehajtjuk a beadott változókat és függvényeket.

```

138     def computeLogger(self, variables, command, names=None, ret=None): # self, variable, command
139         #arg_list = [*args]
140         #variable = arg_list[0]
141         #command = arg_list[1]
142         for variable, name in zip(variables, names):
143             last_read_data = variable[-1]
144             #var_name = f'{variable}={name}'.partition('=')[0]
145             #var_name = [ i for i, a in locals().items() if a == variable][0]
146             #var_name = str(arg_list[0])
147             self.get_logger().info(f'Last read data of {name}: "{last_read_data}"')
148             self.get_logger().info(f'Data of {name}: "{variable}"')
149             avg_of_read_data = np.average(variable)
150             self.get_logger().info(f'Avg of read data: "{avg_of_read_data}"')
151             trans_data = command(variables)
152             self.get_logger().info(f'Trans data of {name}: "{trans_data}"')
153             self.get_logger().info('-----')
154
155         if ret != None:
156             self.get_logger().info('Response sent!')
157             if ret._len_() > 1:
158                 for i in range(ret._len_()):
159                     ret[i][0](ret[i][1], ret[i][2], trans_data[i])
160             else:
161                 ret[0][0](ret[0][1], ret[0][2], trans_data)

```

A következő funkció a robot fejének elfordulásából és a szenzor távolságából számolja ki az irányokat, illetve sebességeket.

```

164     def ul_raw_to_real(self, raw):
165         return int(myMap(raw, 0.0, 4096.0, 0.0, 255.0))
166
167     def get_degrees_and_distance(self, variables):
168         neg_low = -180
169         neg_high = -1
170         pos_low = 1
171         pos_high = 180
172         ul_min = 0
173         ul_max = 25
174         d_v = 0
175         ul_real = self.ul_raw_to_real(variables[1][-1])
176         low_b = 1 if variables[0][-1] > neg_low and variables[0][-1] < neg_high else 0
177         high_b = 1 if variables[0][-1] > pos_low and variables[0][-1] < pos_high else 0
178         d_v = 1 if ul_real > ul_min and ul_real < ul_max else 0
179         l_n_v = 1 if d_v and low_b else 0
180         h_n_v = 1 if d_v and high_b else 0
181         i = l_n_v - h_n_v
182         if i == 1:
183             #return "Left_Back_at_"+str(variables[0][-1])+"_speed"
184             return (1, variables[0][-1]-30.0, ul_real)
185         elif i == -1:
186             #return "Right_Back_at_"+str(variables[0][-1])+"_speed"
187             return (-1, variables[0][-1]-30.0, ul_real)
188         elif i == 0:
189             #return "Back_at_"+str(variables[0][-1])+"_speed"
190             return (0, variables[0][-1]-30.0, ul_real)
191         else:
192             #return "Straight"
193             return (3, 100.0, ul_real)

```

A compute_data függvény minden 0.2 másodpercenként feldolgozza az adatokat, és lehetőség szerint visszaküldi őket.

```

197
198     def compute_data(self):
199         #last_read_data_as = self._as[-1]
200         #self.get_logger().info(f'Last read data: "{last_read_data_as}"')
201         #avg_of_last_read_data_as = np.average(self._as)
202         #trans_data = (now_data*2)
203         #self.get_logger().info(f'Avg of read data: "{avg_of_last_read_data_as}", len: "{self._as.__len__()}"')
204         #self.get_logger().info(f'Last 10 item:{self._as}')
205
206         self.computeLogger(
207             [self._as],
208             lambda x: 1 if x[0][-1] > 50 else 0 if x[0][-1] < -50 else 1,
209             names=["_as"]
210         )
211         self.computeLogger(
212             [self._ts],
213             lambda x: "Pressed" if x[0][-1] else "Released",
214             names=["_ts"]
215         )
216         self.computeLogger(
217             [self._cr, self._cg, self._cb],
218             lambda x: "White" if x[0][-1] > 200 and x[1][-1] > 200 and x[2][-1] > 200 else "Black" \
219             | if x[0][-1] < 50 and x[1][-1] < 50 and x[2][-1] < 50 else "Blue",
220             names=["R", "G", "B"]
221         )
222         self.computeLogger(
223             [self._cd, self._ul],
224             self.get_degrees_and_distance,
225             names=["Head"],
226             ret=[[self.sendPub, self.pub_01, Int8],
227                  [self.sendPub, self.pub_02, Float64],
228                  [self.sendPub, self.pub_03, Int32]
229             ]
230         )

```

```

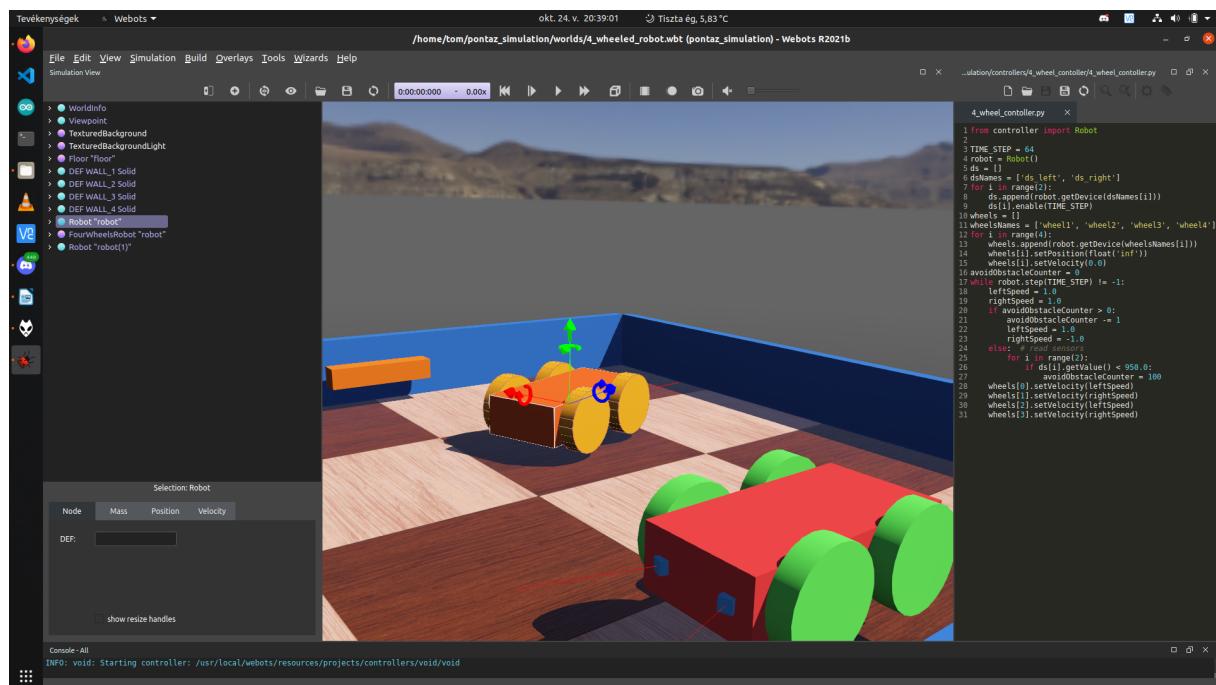
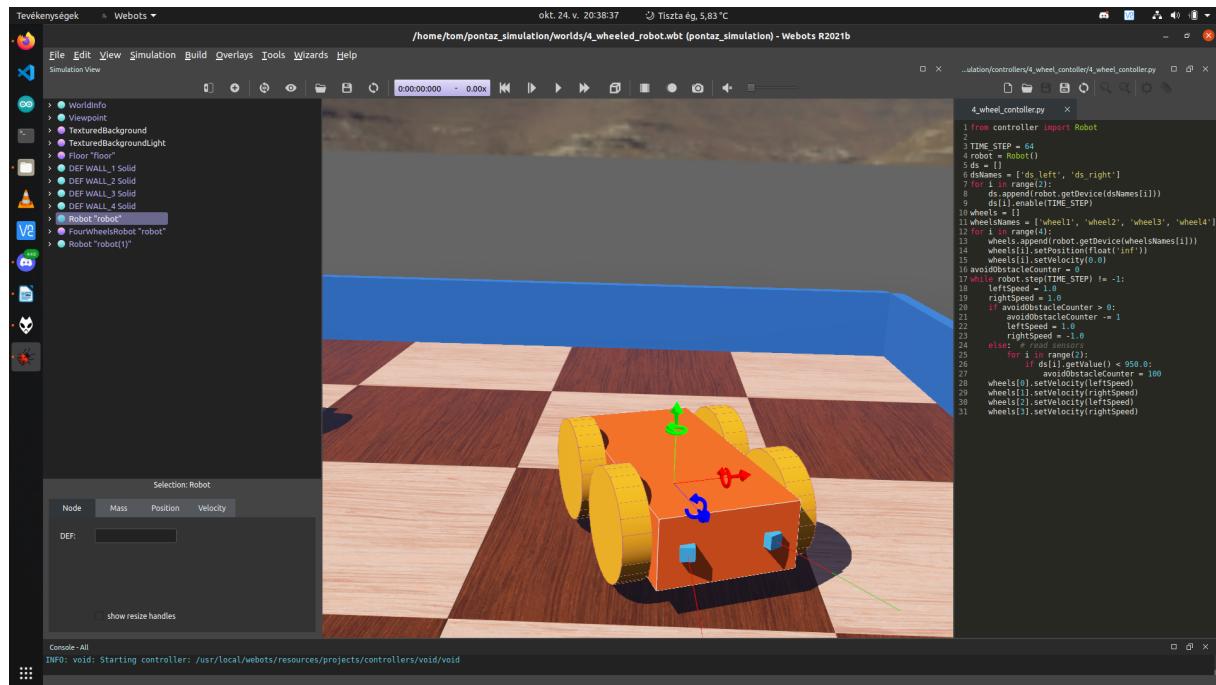
237
238     def main(args=None):
239         rclpy.init(args=args)
240
241         pid = Pid()
242         rclpy.spin(pid)
243
244         # Destroy the node explicitly
245         # (optional - otherwise it will be done automatically
246         # when the garbage collector destroys the node object)
247         pid.destroy_node()
248         rclpy.shutdown()
249
250
251     if __name__ == '__main__':
252         main()

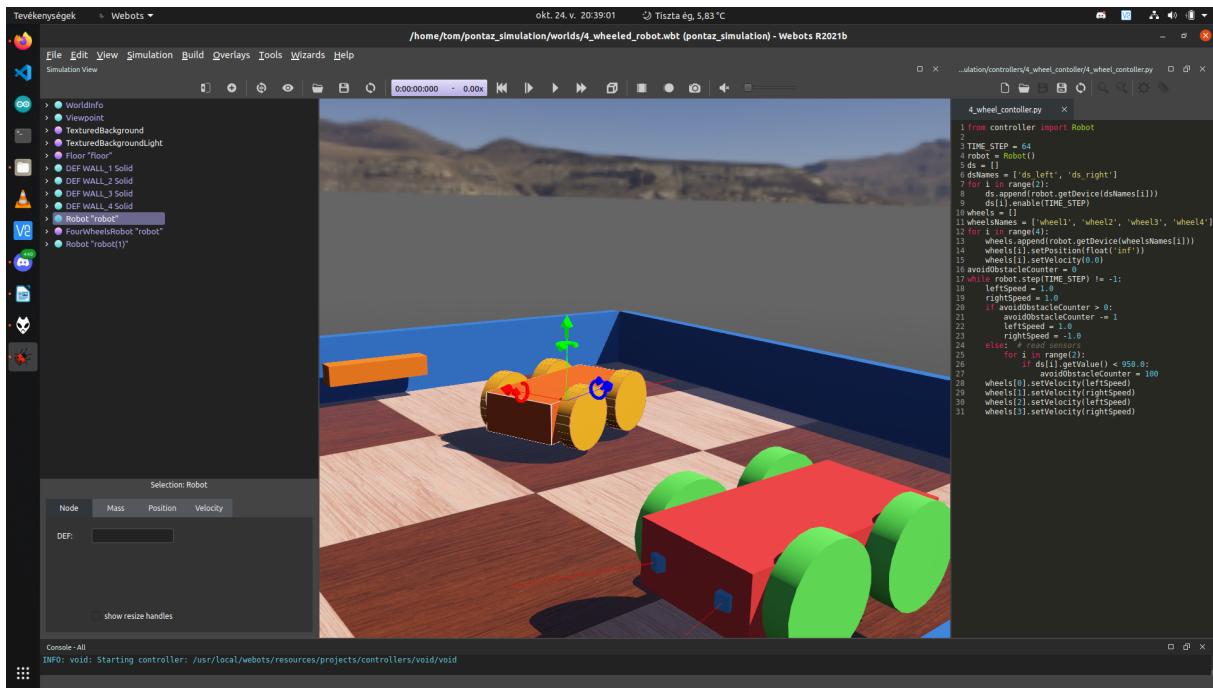
```

2.Feladat:

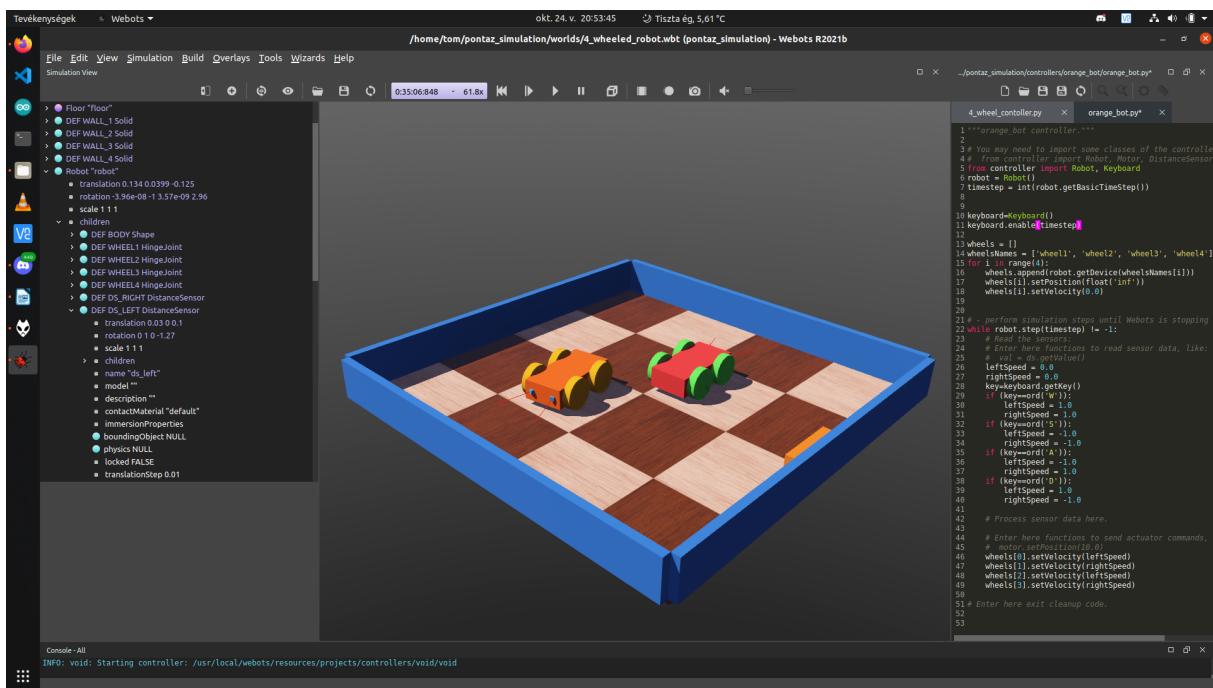
A második feladat egy robot megvalósítása volt Webots-ban.

Képek az elkészült robotról:





A piros-zöld robot 2 távolságmérővel van ellátva, ha falhoz, vagy egyéb objektumhoz közelít, akkor hátrál és megfordul. A narancs-sárga robot az a,w,s,d gombokkal irányítható.



[Link a Webots PROTO-hoz](#)