

- This lesson will show you different ways to control your robot remotely.
- We will start by using keys on the keyboard.
- We will then make a graphical user interface with buttons.
- We will use the **rover7.py** robot program where you set the speed of each wheel.
- Make sure you have PyQt6 installed.
 - Open a power shell or terminal and enter the command below
 - `pip3 install pyqt6`

- On your laptop, open the Python IDLE and enter the program to the right. Call it **idle9.py**.
- Notice the **super().__init__()** function call is the first line in the **__init__()** function. This is used by the system so this class inherits everything from the parent.
- The **__init__()** function creates a UDP socket and sets variable **addr** to the address and port for the robot.

```
import sys
from PyQt6.QtWidgets import *
from PyQt6.QtCore import *
import socket

class Keys(QWidget):
    def __init__(self):
        super().__init__()
        print("Press Keys")
        self.sk = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.addr = ('192.168.4.1', 10000)

    def keyPressEvent(self, event):
        if not event.isAutoRepeat():
            k = event.key()
            if k == Qt.Key.Key_W:
                self.sk.sendto(b'20 20\n', self.addr)
            elif k == Qt.Key.Key_A:
                self.sk.sendto(b'-20 20\n', self.addr)
            elif k == Qt.Key.Key_D:
                self.sk.sendto(b'20 -20\n', self.addr)
            elif k == Qt.Key.Key_S:
                self.sk.sendto(b'-20 -20\n', self.addr)

    def keyReleaseEvent(self, event):
        if not event.isAutoRepeat():
            self.sk.sendto(b'0 0\n', self.addr)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    win = Keys()
    win.show()
    app.exec()
```

- The function **keyPressEvent()** is a standard PyQt6 function to handle all keyboard events in the program. It must have this specific name. It is automatically called when a key is pressed.
- The first thing is the key event is checked if the event is an auto-repeat event. If you press and hold a key down, the computer will start repeating that key. This function detects it. We want to avoid repeating keys and just handle the first press.
- The event is copied to variable k. Then it is checked against four keys.
- The W key will be used to move the robot forward.
- The A key will turn the robot left.
- The S key will turn the robot in reverse.
- The D key will send the robot right.

```
import sys
from PyQt6.QtWidgets import *
from PyQt6.QtCore import *
import socket

class Keys(QWidget):
    def __init__(self):
        super().__init__()
        print("Press Keys")
        self.sk = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.addr = ('192.168.4.1', 10000)

    def keyPressEvent(self, event):
        if not event.isAutoRepeat():
            k = event.key()
            if k == Qt.Key.Key_W:
                self.sk.sendto(b'20 20\n', self.addr)
            elif k == Qt.Key.Key_A:
                self.sk.sendto(b'-20 20\n', self.addr)
            elif k == Qt.Key.Key_D:
                self.sk.sendto(b'20 -20\n', self.addr)
            elif k == Qt.Key.Key_S:
                self.sk.sendto(b'-20 -20\n', self.addr)

    def keyReleaseEvent(self, event):
        if not event.isAutoRepeat():
            self.sk.sendto(b'0 0\n', self.addr)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    win = Keys()
    win.show()
    app.exec()
```

- The **keyReleaseEvent()** is also a standard function that is automatically called when a key is released.
- It checks if the release is from a repeating key. If not, it tells the robot to stop moving.
- The last part of the code includes an if statement. The if statement checks if the program is being run as standalone or was imported into another program. If it is standalone, meaning you are running this program by itself, the lines after the if statement will be executed and the window will open and the program will run.
- If the program is imported, the class is used as a widget in another program.

```
import sys
from PyQt6.QtWidgets import *
from PyQt6.QtCore import *
import socket

class Keys(QWidget):
    def __init__(self):
        super().__init__()
        print("Press Keys")
        self.sk = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.addr = ('192.168.4.1', 10000)

    def keyPressEvent(self, event):
        if not event.isAutoRepeat():
            k = event.key()
            if k == Qt.Key.Key_W:
                self.sk.sendto(b'20 20\n', self.addr)
            elif k == Qt.Key.Key_A:
                self.sk.sendto(b'-20 20\n', self.addr)
            elif k == Qt.Key.Key_D:
                self.sk.sendto(b'20 -20\n', self.addr)
            elif k == Qt.Key.Key_S:
                self.sk.sendto(b'-20 -20\n', self.addr)

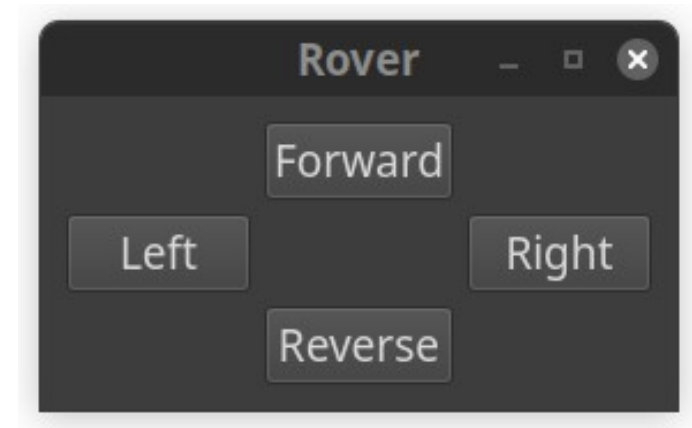
    def keyReleaseEvent(self, event):
        if not event.isAutoRepeat():
            self.sk.sendto(b'0 0\n', self.addr)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    win = Keys()
    win.show()
    app.exec()
```

- Run the rover7.py program on the robot.
- On the laptop, locate the networking icon on the task bar and select the WiFi SSID used for the robot and connect to it.
- Once connected, run the python program.
- Click on the blank window that pops up so it is active.
- Press the W, A, S, D keys and drive the robot around.
- If you want to use the arrow keys on the keyboard,
 - replace the Key_W with Key_Up
 - Key_A with Key_Left
 - Key_D with Key_Right
 - Key_S with Key_Down.

Driving the Robot via WiFi

- This next program will create a simple four button graphical user interface to drive the robot around.
- Create a new file in Python IDLE and call it **idle9a.py**.
- The four buttons will control the direction the robot moves.
- This time, the program will be created as a class object. This will allow easier expansion of the program and makes it more modular.
- The class requires an initialization function. When the class is instantiated or called, the initialization function automatically runs.



Driving the Robot via WiFi

- All the modules that need importing are imported at the top before the class object.
- Next, the class object named Buttons is declared.
- The first function is the initialization function. It must be named as shown.
- You will notice that the socket object `s` has `self.` in front. `self` is a reference to the current instance of the class. It is used in the class to reference any object that include `self`. You can think of it as allowing global access to any functions, variables, and objects inside the class.
- If `self.` is not used, the object or variable is only available in the function it is declared. Other functions in the class cannot access it.
- Since the socket will be needed by other functions, it is declared as **`self.sk`**. This will allow other functions to use it.
- Nothing else in **`__init__()`** needs to be accessed by other functions.

```
import sys
from PyQt6.QtWidgets import *
from PyQt6.QtCore import *
import socket

class Buttons(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Rover')
        fwd = QPushButton('Forward')
        left = QPushButton('Left')
        right = QPushButton('Right')
        rev = QPushButton('Reverse')
        layout = QGridLayout()
        layout.addWidget(fwd, 0, 1)
        layout.addWidget(left, 1, 0)
        layout.addWidget(right, 1, 2)
        layout.addWidget(rev, 2, 1)
        self.setLayout(layout)
        self.sk = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.addr = ('192.168.4.1', 10000)
        fwd.pressed.connect(self.forward)
        fwd.released.connect(self.stop)
        left.pressed.connect(self.left)
        left.released.connect(self.stop)
        right.pressed.connect(self.right)
        right.released.connect(self.stop)
        rev.pressed.connect(self.reverse)
        rev.released.connect(self.stop)
```

Laptop Program

Driving the Robot via WiFi

- The window title will be changed to Rover.
- Four buttons are declared.
- The layout used is the grid. The buttons are positioned using the row and column values.
- The UDP socket is declared. It has self. in front so it can be used in other functions in the class.
- Lastly, the buttons are connected to events. The pressed events are connected to functions that make the robot move in the desired direction.
- The released events are connected to the stop function to stop the robot movement.

```
import sys
from PyQt6.QtWidgets import *
from PyQt6.QtCore import *
import socket

class Buttons(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Rover')
        fwd = QPushButton('Forward')
        left = QPushButton('Left')
        right = QPushButton('Right')
        rev = QPushButton('Reverse')
        layout = QGridLayout()
        layout.addWidget(fwd, 0, 1)
        layout.addWidget(left, 1, 0)
        layout.addWidget(right, 1, 2)
        layout.addWidget(rev, 2, 1)
        self.setLayout(layout)
        self.sk = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.addr = ('192.168.4.1', 10000)
        fwd.pressed.connect(self.forward)
        fwd.released.connect(self.stop)
        left.pressed.connect(self.left)
        left.released.connect(self.stop)
        right.pressed.connect(self.right)
        right.released.connect(self.stop)
        rev.pressed.connect(self.reverse)
        rev.released.connect(self.stop)
```

Laptop Program

Driving the Robot via WiFi

- Each function simply sends the motor speed to the robot to perform the desired motion.
- And at the bottom is the standard code to run the program.
- Run rover7.py on the robot and then run this program. Click on the buttons drive the robot around.

```
def stop(self):  
    self.sk.sendto(b'0 0\n', self.addr)  
  
def forward(self):  
    self.sk.sendto(b'20 20\n', self.addr)  
  
def reverse(self):  
    self.sk.sendto(b'-20 -20\n', self.addr)  
  
def left(self):  
    self.sk.sendto(b'-20 20\n', self.addr)  
  
def right(self):  
    self.sk.sendto(b'20 -20\n', self.addr)  
  
if __name__ == '__main__':  
    app = QApplication(sys.argv)  
    win = Buttons()  
    win.show()  
    app.exec()
```

Laptop Program

- We can let the program handle both buttons and keys to control the robot.
- Copy the **keyPressEvent()** and **keyReleaseEvent()** functions to the **buttons.py** program. Use the letter keys as the arrow keys are used to navigate and select the buttons.
- Try the program again and verify both the buttons and keys work. Just do not use the arrow keys. They will not work.

