



Instituto Tecnológico Nacional de México  
Instituto Tecnológico de Iztapalapa



## ***Ingeniería Mecatrónica***

***Materia: Programación Avanzada***

***Grupo M7A***

***Examen de recuperación***

***Alumno: Navarrete Torres Mauricio Stanhí 201080073***

***Profesor: Soria Frias Sigfrido Oscar***

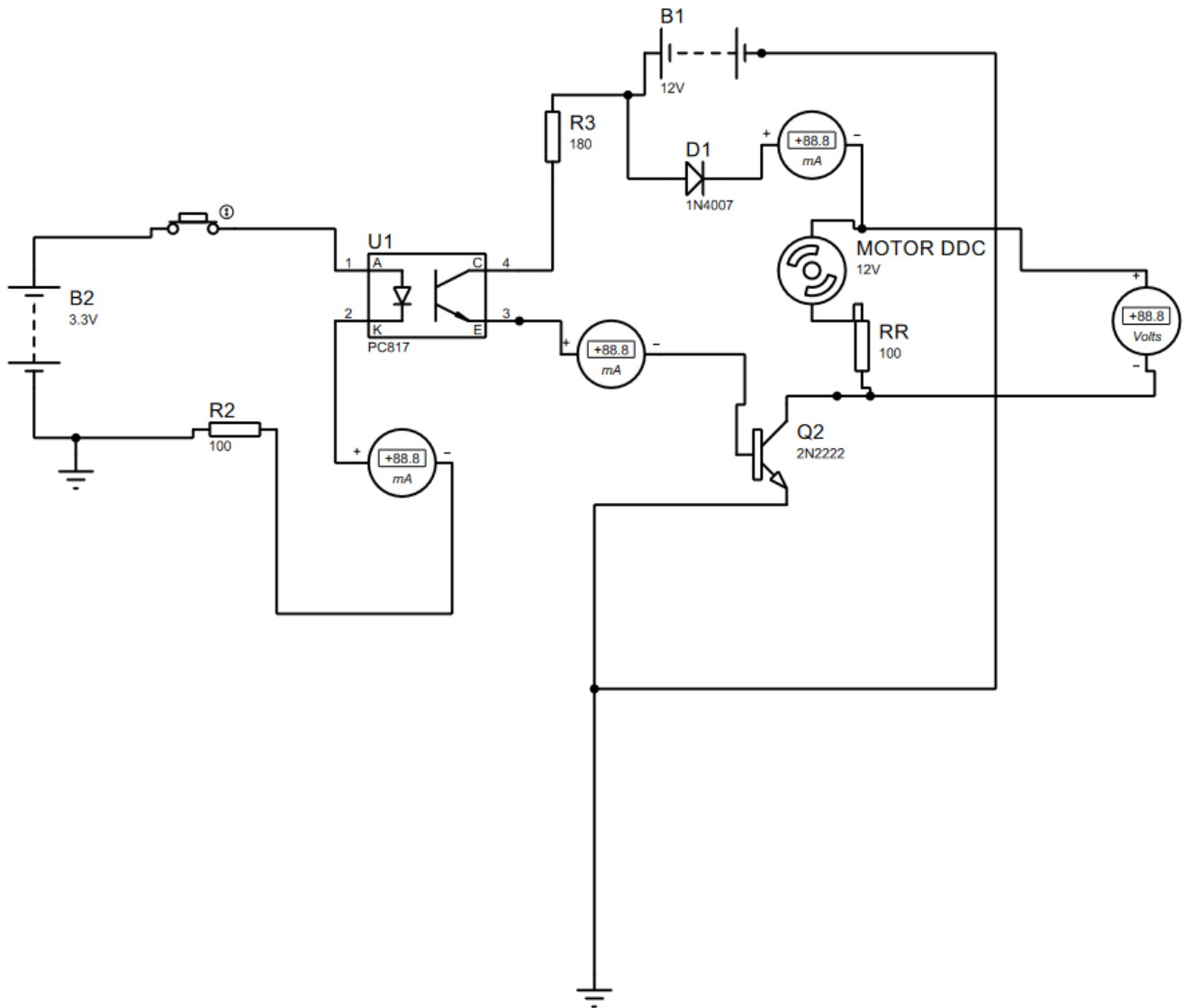


## Introducción

En el presente documento se presenta el programa requerido en el examen tipo “B”

## Electrónica

La electrónica utilizada para ambas salidas:



## código

A continuación el código

```
import sys

from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QPushButton, QLabel
from PyQt5.QtCore import QTimer
from gpiozero import Button, LED, OutputDevice
from threading import Thread

import time

# Configuración de GPIOs
hs01 = Button(2) # GPIO 2 para HS-01
hs02 = Button(3) # GPIO 3 para HS-02
m1 = OutputDevice(17) # GPIO 17 para motor M1
pl01 = LED(27) # GPIO 27 para foco LED PL-01

# Clase principal de la interfaz gráfica
class MainWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Controlador de Raspberry Pi")
        self.layout = QVBoxLayout()

        # Estado inicial de los botones y dispositivos
        self.hs03_active = False
        self.hs01_active = False
        self.hs02_active = False
        self.m1_active = False
```

```
self.pl01_active = False

# Botón virtual enclavado HS-03
self.hs03_button = QPushButton("HS-03 (Desactivado)")
self.hs03_button.setCheckable(True)
self.hs03_button.clicked.connect(self.toggle_hs03)
self.layout.addWidget(self.hs03_button)

# Etiquetas para mostrar el estado de los dispositivos
self.hs01_label = QLabel("HS-01: Desactivado")
self.hs02_label = QLabel("HS-02: Desactivado")
self.m1_label = QLabel("M1: Desactivado")
self.pl01_label = QLabel("PL-01: Desactivado")

self.layout.addWidget(self.hs01_label)
self.layout.addWidget(self.hs02_label)
self.layout.addWidget(self.m1_label)
self.layout.addWidget(self.pl01_label)

self.setLayout(self.layout)

# Temporizadores para actualizar la interfaz gráfica
self.timer = QTimer()
self.timer.timeout.connect(self.update_status)
self.timer.start(100) # Actualizar cada 100 ms

# Hilo para el parpadeo del foco LED
self.led_thread = Thread(target=self.blink_led)
self.led_thread.start()
```

```
# Eventos de botones físicos

hs01.when_pressed = self.hs01_pressed
hs02.when_pressed = self.hs02_pressed
hs02.when_released = self.hs02_released
hs01.when_released = self.hs01_released


def toggle_hs03(self):
    self.hs03_active = not self.hs03_active
    if self.hs03_active:
        self.hs03_button.setText("HS-03 (Activado)")
    else:
        self.hs03_button.setText("HS-03 (Desactivado)")


def hs01_pressed(self):
    self.hs01_active = True
    if self.hs03_active:
        self.activate_motor(5)
    else:
        self.activate_motor(10)


def hs01_released(self):
    self.hs01_active = False


def hs01_status(self):
    if self.hs01_active:
        if self.hs03_active:
            self.activate_motor(5)
        else:
            self.activate_motor(10)
```

```

def hs02_pressed(self):
    self.hs02_active = True

def hs02_released(self):
    self.hs02_active = False

def activate_motor(self, duration):
    self.m1_active = True
    m1.on()
    time.sleep(duration)
    m1.off()
    self.m1_active = False

def blink_led(self):
    while True:
        pl01.on()
        self.pl01_active = True
        time.sleep(3)
        pl01.off()
        self.pl01_active = False
        time.sleep(1)

def update_status(self):
    # Actualizar etiquetas según el estado de los dispositivos
    self.hs01_label.setText(f"HS-01: {'Activado' if self.hs01_active else 'Desactivado'}")
    self.hs02_label.setText(f"HS-02: {'Activado' if self.hs02_active else 'Desactivado'}")
    self.m1_label.setText(f"M1: {'Activado' if self.m1_active else 'Desactivado'}")

```

```
        self.pl01_label.setText(f"PL-01: {'Activado' if self.pl01_active else  
'Desactivado'}")  
  
        # Colorear las etiquetas según el estado  
        self.hs01_label.setStyleSheet(f"color: {'green' if self.hs01_active else  
'red'}")  
        self.hs02_label.setStyleSheet(f"color: {'green' if self.hs02_active else  
'red'}")  
        self.m1_label.setStyleSheet(f"color: {'green' if self.m1_active else  
'red'}")  
        self.pl01_label.setStyleSheet(f"color: {'green' if self.pl01_active else  
'red'}")  
  
# Ejecución de la aplicación  
if __name__ == "__main__":  
    app = QApplication(sys.argv)  
    window = MainWindow()  
    window.show()  
    sys.exit(app.exec_())
```