

Dependències del Sistema de Control de Robot

Aquest document detalla totes les dependències necessàries per executar correctament el Sistema de Control de Robot en un nou ordinador, i també inclou informació sobre els elements pendents d'implementació.

Dependències Principals

Instal·leu-les amb `pip install [paquet]` o utilitzeu l'arxiu `requirements.txt` proporcionat.

 Copiar

```
# Per instal·lar totes les dependències d'un sol cop:  
pip install -r requirements.txt
```

Llibreries bàsiques

- **PyQt5:** Interfície gràfica d'usuari

 Copiar

```
pip install PyQt5
```

- **NumPy:** Càlculs numèrics i manipulació d'arrays

 Copiar

```
pip install numpy
```

- **SciPy:** Funcionalitats científiques i interpolació

 Copiar

```
pip install scipy
```

- **Matplotlib:** Visualització de dades

 Copiar

```
pip install matplotlib
```

- **Logging:** Part de la biblioteca estàndard de Python, no requereix instal·lació addicional

Llibreries per a processament d'imatge (per al mòdul de càmera)

- **OpenCV:** Processament d'imatge i visió per computador

 Copiar

```
pip install opencv-python
```

Contingut per al fitxer requirements.txt

Podeu crear un fitxer `requirements.txt` amb el següent contingut per facilitar la instal·lació:

 Copiar

```
PyQt5>=5.15.0  
numpy>=1.19.0  
scipy>=1.5.0  
matplotlib>=3.3.0  
opencv-python>=4.5.0
```

Instal·lació en diferents sistemes operatius

Windows

1. Instal·leu Python (versió 3.8 o superior recomanada):

- Descarregueu-lo des de [python.org](https://www.python.org)
- Assegureu-vos de marcar l'opció "Add Python to PATH" durant la instal·lació

2. Obriu el Command Prompt i instal·leu les dependències:

 Copiar

```
pip install -r requirements.txt
```

Linux (Ubuntu/Debian)

1. Instal·leu Python i pip:

 Copiar

```
sudo apt update  
sudo apt install python3 python3-pip python3-dev
```

2. Instal·leu les dependències del sistema necessàries per a PyQt5:

 Copiar

```
sudo apt install python3-pyqt5 python3-pyqt5.qtsvg
```

3. Instal·leu les dependències Python:

 Copiar

```
pip3 install -r requirements.txt
```

macOS

1. Instal·leu Homebrew si encara no el teniu:

 Copiar

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.bash)"
```

2. Instal·leu Python:

 Copiar

```
brew install python
```

3. Instal·leu les dependències Python:

 Copiar

```
pip3 install -r requirements.txt
```

Notes addicionals

- El mòdul de càmera requereix OpenCV, però el sistema funcionarà amb funcionalitats limitades si no està instal·lat.
- Per a una visualització més avançada, assegureu-vos que Matplotlib està correctament instal·lat.
- Si teniu problemes amb PyQt5, podeu provar d'instal·lar-lo des de la pàgina oficial o utilitzar conda per a la seva instal·lació.

Estructura de directoris a crear

Els següents directoris i fitxers són necessaris per al correcte funcionament de l'aplicació, però alguns no estan completament implementats en el codi actual:

```

robot_control_system/
|
|   └── resources/          # Recursos estàtics (parcialment implementat)
|       ├── styles/          # Fulls d'estil QSS (no implementat)
|       |   └── main.qss      # Full d'estils principal (pendent de crear)
|       └── icons/           # Ícones
|           ├── robot_icon.png # Ícone de l'aplicació
|           └── splash.png    # Imatge de benvinguda
|
|   └── config/             # Directori per a configuracions (parcialment implementat)
|       ├── config.ini       # Fitxer de configuració principal
|       └── profiles/        # Perfil de configuració (no implementat)
|
|   └── logs/               # Directori per a logs (parcialment implementat)
|
|   └── db/                 # Directori per a base de dades (no implementat)
|       └── projects.db      # Base de dades de projectes (pendent de crear)

```

Funcionalitats pendents d'implementar

Les següents funcionalitats estan previstes en el codi però no s'han implementat completament:

- 1. Estilització amb QSS:** El sistema està preparat per carregar fulls d'estil QSS (`main.qss`), però encara no s'han creat.
- 2. Perfil de configuració:** Existeix la infraestructura per gestionar perfils de configuració al directori `config/profiles/`, però no està completament implementat.
- 3. Base de dades per a projectes:** No s'ha implementat una base de dades per emmagatzemar informació de projectes.
- 4. Visualització del LiDAR:** El codi preveu visualització en mode cartesià i polar, però no està completament implementat a la interfície.
- 5. Pantalla de Splash:** El codi inclou funcionalitat per mostrar una pantalla de benvinguda `splash.png`, però cal crear-la.

Script d'instal·lació i preparació de l'entorn

Per facilitar la preparació del sistema en un nou ordinador, he creat un script `setup.py` que automatitza tot el procés:

```
#!/usr/bin/env python3
"""

Script de configuració inicial del Sistema de Control de Robot
=====
Aquest script prepara l'entorn per a l'execució del sistema:
1. Verifica i instal·la dependències
2. Crea l'estructura de directoris necessària
3. Genera fitxers de configuració inicials
4. Inicialitza la base de dades
"""

import os
import sys
import subprocess
import shutil

def print_section(title):
    """Imprimeix una secció amb format."""
    print("\n" + "=" * 80)
    print(f" {title} ".center(80, "="))
    print("=". * 80 + "\n")

def install_dependencies():
    """Instal·la les dependències necessàries."""
    print_section("Instal·lació de dependències")

    dependencies = [
        "PyQt5",
        "numpy",
        "scipy",
        "matplotlib",
        "opencv-python"
    ]

    print("Instal·lant dependències...")

    for dep in dependencies:
        print(f"Instal·lant {dep}...")
        try:
            subprocess.check_call([sys.executable, "-m", "pip", "install", dep])
            print(f"✓ {dep} instal·lat correctament")
        except subprocess.CalledProcessError:
            print(f"✗ Error instal·lant {dep}")
            return False

    return True
```

```

print("\nTotes les dependències s'han instal·lat correctament!")
return True

def create_directory_structure():
    """Crea l'estructura de directoris necessària."""
    print_section("Creació d'estructura de directoris")

    directories = [
        "resources",
        "resources/styles",
        "resources/icons",
        "config",
        "config/profiles",
        "logs",
        "db"
    ]

    for directory in directories:
        if not os.path.exists(directory):
            print(f"Creant directori {directory}...")
            os.makedirs(directory, exist_ok=True)
        else:
            print(f"El directori {directory} ja existeix.")

    print("\nEstructura de directoris creada correctament!")
    return True

def create_basic_files():
    """Crea els fitxers bàsics necessaris."""
    print_section("Creació de fitxers bàsics")

    # Contingut del fitxer main.qss
    main_qss_content = """/* Full d'estils principal del Sistema de Control de Robot */

/* Estil general de l'aplicació */
QWidget {
    font-family: Arial, sans-serif;
    font-size: 12px;
    background-color: #f0f0f0;
}

QMainWindow {
    background-color: #e6e6e6;
}

/* Estils per botons */
"""

```

```
QPushButton {  
    background-color: #2a82da;  
    color: white;  
    border: 1px solid #1e6cbf;  
    border-radius: 4px;  
    padding: 5px 10px;  
}  
  
QPushButton:hover {  
    background-color: #3a92ea;  
}  
  
QPushButton:pressed {  
    background-color: #1e6cbf;  
}  
  
/* Estils per a panells de control */  
QGroupBox {  
    border: 1px solid #999999;  
    border-radius: 5px;  
    margin-top: 20px;  
    font-weight: bold;  
}  
  
QGroupBox::title {  
    subcontrol-origin: margin;  
    subcontrol-position: top center;  
    padding: 0 10px;  
}  
  
/* Estils per a QSlider */  
QSlider::groove:horizontal {  
    border: 1px solid #999999;  
    height: 8px;  
    background: #cccccc;  
    margin: 2px 0;  
}  
  
QSlider::handle:horizontal {  
    background: #2a82da;  
    border: 1px solid #1e6cbf;  
    width: 18px;  
    margin: -2px 0;  
    border-radius: 3px;  
}  
....
```

```
# Contingut del fitxer config.ini
config_ini_content = """[connection]
host=192.168.1.100
port=9999
auto_reconnect=true
reconnect_interval=5

[lidar]
enabled=true
scan_frequency=5
max_distance=3000

[camera]
enabled=true
resolution=640,480
fps=15

[navigation]
default_speed=150
auto_stop_timeout=30
obstacle_threshold=500

[ui]
theme=light
language=ca
sound_alerts=true
show_statistics=true
"""

files_to_create = [
    ("resources/styles/main.qss", main_qss_content),
    ("config/config.ini", config_ini_content)
]

for file_path, content in files_to_create:
    if not os.path.exists(file_path):
        print(f"Creant fitxer {file_path}...")
        with open(file_path, 'w', encoding='utf-8') as f:
            f.write(content)
    else:
        print(f"El fitxer {file_path} ja existeix.")

print("\nFitxers bàsics creats correctament!")
return True

def initialize_database():
    """Inicialitza la base de dades."""
```

```
print_section("Inicialització de la base de dades")

# Crear directori de la base de dades si no existeix
if not os.path.exists("db"):
    os.makedirs("db", exist_ok=True)

# Comprovar si la base de dades ja existeix
if os.path.exists("db/projects.db"):
    print("La base de dades ja existeix. S'ha de recrear? (s/n)")
    choice = input().lower()
    if choice == 's':
        os.remove("db/projects.db")
    else:
        print("S'utilitzarà la base de dades existent.")
        return True

try:
    import sqlite3

    # Crear la base de dades i les taules
    conn = sqlite3.connect("db/projects.db")
    cursor = conn.cursor()

    # Crear taula de projectes
    cursor.execute('''
CREATE TABLE IF NOT EXISTS projects (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
'''')

    # Crear taula de configuracions
    cursor.execute('''
CREATE TABLE IF NOT EXISTS configurations (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    project_id INTEGER,
    name TEXT NOT NULL,
    config_data TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (project_id) REFERENCES projects (id)
)
'''')

    # Crear projecte d'exemple

```

```
cursor.execute(
    "INSERT INTO projects (name, description) VALUES (?, ?)",
    ("Projecte de demostració", "Projecte inicial creat automàticament per a demostració")
)

conn.commit()
conn.close()

print("Base de dades inicialitzada correctament!")
return True

except Exception as e:
    print(f"Error inicialitzant la base de dades: {e}")
    return False

def main():
    """Funció principal."""
    print_section("Configuració del Sistema de Control de Robot")

    print("Aquest script configurarà l'entorn per al Sistema de Control de Robot.")
    print("S'instal·laran les dependències necessàries i es crearà l'estructura de directoris.")
    print("\nVoleu continuar? (s/n)")
    choice = input().lower()

    if choice != 's':
        print("Configuració cancel·lada.")
        return

    # Passos de configuració
    steps = [
        ("Instal·lació de dependències", install_dependencies),
        ("Creació d'estructura de directoris", create_directory_structure),
        ("Creació de fitxers bàsics", create_basic_files),
        ("Inicialització de la base de dades", initialize_database)
    ]

    for step_name, step_func in steps:
        print(f"\nExecutant: {step_name}...")
        if not step_func():
            print(f"Error durant {step_name}. S'atura la configuració.")
            return

    print_section("Configuració completada")
    print("El Sistema de Control de Robot s'ha configurat correctament.")
    print("Podeu executar l'aplicació amb: python main.py")
```

```
if __name__ == "__main__":
    main()
```

Verificació de la instal·lació

Si preferiu verificar manualment les dependències, podeu utilitzar aquest script:

```
# verify_dependencies.py
import sys
import os

# Comprovar les dependències de Python
dependencies = {
    "PyQt5": ["QtCore", "QtGui", "QtWidgets"],
    "numpy": None,
    "scipy": ["interpolate"],
    "matplotlib": ["pyplot"],
    "cv2": None
}

missing = []

for package, modules in dependencies.items():
    try:
        imported_package = __import__(package)
        if modules:
            for module in modules:
                try:
                    __import__(f"{package}.{module}")
                except ImportError:
                    missing.append(f"{package}.{module}")
    print(f"✓ {package}")
except ImportError:
    missing.append(package)
print(f"✗ {package}")

# Verificar estructura de directoris
required_dirs = [
    "resources",
    "resources/styles",
    "resources/icons",
    "config",
    "config/profiles",
    "logs",
    "db"
]

for directory in required_dirs:
    if os.path.exists(directory):
        print(f"✓ Directori '{directory}'")
    else:
```

```

print(f"\x Directori '{directory}' (falta crear-lo)")
try:
    os.makedirs(directory, exist_ok=True)
    print(f" - Directori '{directory}' creat automàticament")
except Exception as e:
    print(f" - No s'ha pogut crear el directori '{directory}': {e}")

# Verificar fitxers critics
required_files = [
    ("resources/styles/main.qss", "/* Full d'estils principal */\n/* Estil global de l'aplica"),
    ("config/config.ini", "[connection]\nhost=192.168.1.100\nport=9999\nauto_reconnect=true\nre"),
]

for file_path, default_content in required_files:
    if os.path.exists(file_path):
        print(f"\x Fitxer '{file_path}'")
    else:
        print(f"\x Fitxer '{file_path}' (falta crear-lo)")
        try:
            with open(file_path, 'w', encoding='utf-8') as f:
                f.write(default_content)
            print(f" - Fitxer '{file_path}' creat automàticament amb contingut per defecte")
        except Exception as e:
            print(f" - No s'ha pogut crear el fitxer '{file_path}': {e}")

if missing:
    print("\nDependències que falten:")
    for package in missing:
        print(f" - {package}")
    print("\nInstal·leu les dependències que falten amb pip abans d'executar l'aplicació.")
else:
    print("\nTotes les dependències de Python estan correctament instal·lades!")

```

Executeu aquest script amb `python verify_dependencies.py` per comprovar que tot està correctament instal·lat i crear automàticament els directoris i fitxers bàsics necessaris.