DonKeWu / **Futterwagen2.0**

Code Issues Pull requests Actions Projects Wiki Secur

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.

base: main ... compare: master

There isn't anything to compare.

main and **master** are entirely different commit histories.

Showing **11 changed files** with **166 additions** and **10 deletions**. Split **Unified**

1 conrollers/__init__.py

```
...  ...  @@ -0,0 +1 @@
     1  + from .fuetterung_controller import FütterungController
```

23 conrollers/fuetterung_controller.py

```
...  ...  @@ -0,0 +1,23 @@
     1  + from typing import List
     2  + from models import Fütterung, Pferd, Futter
     3  + from datetime import datetime
     4  +
     5  + class FütterungController:
     6  +     def __init__(self):
     7  +         self.fütterungen: List[Fütterung] = []
     8  +
     9  +     def neue_fütterung(self, pferd: Pferd, futter: Futter, menge_kg:
          float, zeitpunkt: datetime):
    10  +         # Dummy-Nährwertberechnung
    11  +         naehrwerte = {
    12  +             "Rohprotein": futter.rohprotein * menge_kg,
    13  +             "Rohfaser": futter.rohfaser * menge_kg,
    14  +         }
    15  +         fütterung = Fütterung(
```

```
16 +             pferd=pferd,
17 +             futter=futter,
18 +             menge_kg=menge_kg,
19 +             naehrwerte=naehrwerte,
20 +             zeitpunkt=zeitpunkt
21 +         )
22 +         self.fütterungen.append(fütterung)
23 +         return fütterung
```

1 hardware/__init__.py

```
...  ...  @@ -0,0 +1 @@
     1  + from .sensor_interface import WeightSensorInterface
```

12 hardware/hx711_sensor.py

```
...  ...  @@ -0,0 +1,12 @@
     1  + from .sensor_interface import WeightSensorInterface
     2  +
     3  + class HX711Sensor(WeightSensorInterface):
     4  +     def __init__(self, data_pin: int, clock_pin: int):
     5  +         # Initialisierung der HX711-Hardware (Pseudo-Code)
     6  +         self.data_pin = data_pin
     7  +         self.clock_pin = clock_pin
     8  +
     9  +     def read_weight(self) -> float:
    10  +         # Hier würdest du die echte Bibliothek ansprechen, z.B. hx711python
    11  +         # return hx711.get_weight()
    12  +         return 2.5  # Dummywert für Beispiel
```

6 hardware/sensor_interface.py

```
...  ...  @@ -0,0 +1,6 @@
     1  + from abc import ABC, abstractmethod
     2  +
     3  + class WeightSensorInterface(ABC):
     4  +     @abstractmethod
     5  +     def read_weight(self) -> float:
     6  +         pass
```

39 main.py

```
...  ...  @@ -1,16 +1,35 @@
 1       - # This is a sample Python script.
     1  + import sys
     2  + from datetime import datetime
     3  + from PyQt5.QtWidgets import QApplication
```

```
     4 + from models import Pferd, Heulage
     5 + from controllers import FütterungController
     6 + from hardware.hx711_sensor import HX711Sensor
     7 + from views.main_window import MainWindow
2    8
3      - # Press Umschalt+F10 to execute it or replace it with your code.
4      - # Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.
5    9
    10 + def main():
    11 +     # 1. Hardware initialisieren
    12 +     sensor = HX711Sensor(data_pin=5, clock_pin=6)  # GPIO-Pins anpassen
6   13
7      - def print_hi(name):
8      -     # Use a breakpoint in the code line below to debug your script.
9      -     print(f'Hi, {name}')  # Press Strg+F8 to toggle the breakpoint.
    14 +     # 2. UI starten
    15 +     app = QApplication(sys.argv)
    16 +     window = MainWindow(sensor)
    17 +     window.show()
10  18
    19 +     # 3. Testdaten laden (optional/nur für Entwicklung)
    20 +     if True:  # Auf False setzen für Produktivbetrieb
    21 +         pferd = Pferd(name="Blitz", gewicht=500, alter=8)
    22 +         heulage = Heulage(
    23 +             name="Heulage 2024", trockenmasse=60.0, rohprotein=14.0, rohfaser=24.0,
    24 +             gesamtzucker=8.0, fruktan=4.0, me_pferd=8.0, pcv_xp=7.0,
    25 +             herkunft="Hof B", jahrgang=2024, ph_wert=4.5, siliergrad="gut"
    26 +         )
    27 +         controller = FütterungController()
    28 +         controller.neue_fütterung(pferd, heulage, 2.5, datetime.now())
11  29
12     - # Press the green button in the gutter to run the script.
13     - if __name__ == '__main__':
14     -     print_hi('PyCharm')
    30 +     sys.exit(app.exec_())
    31 +
    32 +
    33 + if __name__ == "__main__":
    34 +     main()
15  35
16     - # See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

▾ 3 models/__init__.py

```
...  ...  @@ -0,0 +1,3 @@
```

```
1  + from .pferd import Pferd
2  + from .futter import Futter, Heu, Heulage, PelletFutter, Hafer
3  + from .fuetterung import Fütterung
```

13 models/fuetterung.py

```
...  ...  @@ -0,0 +1,13 @@
1  + from dataclasses import dataclass
2  + from typing import Optional, Dict
3  + from datetime import datetime
4  + from .pferd import Pferd
5  + from .futter import Futter
6  +
7  + @dataclass
8  + class Fütterung:
9  +     pferd: Pferd
10 +     futter: Futter
11 +     menge_kg: float
12 +     naehrwerte: Optional[Dict[str, float]] = None
13 +     zeitpunkt: Optional[datetime] = None
```

33 models/futter.py

```
...  ...  @@ -0,0 +1,33 @@
1  + from dataclasses import dataclass
2  + from typing import Optional, Dict
3  +
4  + @dataclass
5  + class Futter:
6  +     name: str
7  +     trockenmasse: float
8  +     rohprotein: float
9  +     rohfaser: float
10 +     gesamtzucker: float
11 +     fruktan: float
12 +     me_pferd: float
13 +     pcv_xp: float
14 +     herkunft: Optional[str] = None
15 +     jahrgang: Optional[int] = None
16 +
17 + @dataclass
18 + class Heu(Futter):
19 +     staubarm: Optional[bool] = None
20 +
21 + @dataclass
22 + class Heulage(Futter):
23 +     ph_wert: Optional[float] = None
24 +     siliergrad: Optional[str] = None
25 +
```

```
26 + @dataclass
27 + class PelletFutter(Futter):
28 +     zusatzstoffe: Optional[Dict[str, float]] = None
29 +
30 + @dataclass
31 + class Hafer(Futter):
32 +     sorte: Optional[str] = None
33 +     stärke: Optional[float] = None
```

9 ■■■■■ models/pferd.py

```
...  ...  @@ -0,0 +1,9 @@
 1 + from dataclasses import dataclass
 2 + from typing import Optional
 3 +
 4 + @dataclass
 5 + class Pferd:
 6 +     name: str
 7 +     gewicht: float
 8 +     alter: int
 9 +     notizen: Optional[str] = None
```

36 ■■■■■ views/main_window.py

```
...  ...  @@ -0,0 +1,36 @@
 1 + from PyQt5.QtWidgets import QMainWindow, QLabel, QPushButton,
      QVBoxLayout, QWidget
 2 + from PyQt5.QtCore import QTimer
 3 +
 4 + class MainWindow(QMainWindow):
 5 +     def __init__(self, sensor):
 6 +         super().__init__()
 7 +         self.sensor = sensor
 8 +
 9 +         self.setWindowTitle("Futterkarre 2.0")
10 +         self.setFixedSize(1024, 600)
11 +
12 +         self.weight_label = QLabel("Gewicht: -- kg")
13 +         self.refresh_button = QPushButton("Aktualisieren")
14 +
15 +         layout = QVBoxLayout()
16 +         layout.addWidget(self.weight_label)
17 +         layout.addWidget(self.refresh_button)
18 +
19 +         container = QWidget()
20 +         container.setLayout(layout)
21 +         self.setCentralWidget(container)
22 +
23 +         self.refresh_button.clicked.connect(self.update_weight)
```

```
24 +
25 +         # Automatische Aktualisierung alle 1 Sekunde
26 +         self.timer = QTimer()
27 +         self.timer.timeout.connect(self.update_weight)
28 +         self.timer.start(1000)
29 +
30 +     def update_weight(self):
31 +         try:
32 +             weight = self.sensor.read_weight()
33 +             self.weight_label.setText(f"Gewicht: {weight:.2f} kg")
34 +         except Exception as e:
35 +             self.weight_label.setText("Fehler beim Wiegen!")
36 +             # Hier könntest du ein Logging-Framework nutzen
```