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 **10 changed files** with **344 additions** and **64 deletions**. Split **Unified**

BIN -14.9 KB (85%) Comparing main...master · DonKeWu_Futterwagen2.0.pdf

Binary file not shown.

11 config/logging_config.py

```
...  ...  @@ -1,5 +1,12 @@
 1       - # utils/logging_config.py
      1  + # config/logging_config.py
 2    2    import logging
 3    3
 4    4    def setup_logging():
 5       -     logging.basicConfig(level=logging.INFO, format="%(asctime)s
          [%(levelname)s] %(message)s")
      5  +     logging.basicConfig(
      6  +         level=logging.INFO,
      7  +         format="%(asctime)s [%(levelname)s] %(name)s: %(message)s",
      8  +         handlers=[
      9  +             logging.StreamHandler()
     10  +         ]
     11  +     )
     12  +
```

0 conrollers/__init__.py → controllers/__init__.py

File renamed without changes.

0

conrollers/fuetterung_controller.py → controllers/fuetterung_controller.py

File renamed without changes.

18 main.py

```
...  ...  @@ -1,4 +1,8 @@
 1    1   # main.py
      2  + from config.logging_config import setup_logging
      3  + setup_logging()
      4  + import logging
      5  + logger = logging.getLogger(__name__)
 2    6
 3    7   import sys
 4    8   from datetime import datetime
 7   11   from controllers.fuetterung_controller import FütterungController
 8   12   from hardware.hx711_sensor import HX711Sensor
 9   13   from views.main_window import MainWindow
10       - from utils.futter_loader import lade_heu_aus_csv,
         lade_heulage_aus_csv  # Optional, wenn du CSV laden willst
     14  + from utils.futter_loader import finde_heu_dateien
11   15
12   16   def main():
13   17       # 1. Hardware initialisieren
14   18       sensor = HX711Sensor(data_pin=5, clock_pin=6)  # GPIO-Pins
         anpassen
15   19
16       -     # 2. UI starten
     20  +     # 2. Heu-Dateien aus dem data-Ordner laden
     21  +     heu_namen = finde_heu_dateien()
     22  +
     23  +
     24  +     # 3. PyQt-Anwendung starten
17   25       app = QApplication(sys.argv)
18       -     window = MainWindow(sensor)
     26  +     window = MainWindow(sensor, heu_namen=heu_namen)
19   27       window.show()
20   28
21       -     # 3. Testdaten laden (optional/nur für Entwicklung)
     29  +     # 4. Testdaten laden (optional/nur für Entwicklung)
22   30       if True:  # Auf False setzen für Produktivbetrieb
23   31           pferd = Pferd(name="Blitz", gewicht=500, alter=8)
24   32           heulage = Heulage(
```

```
33  41
34  42    if __name__ == "__main__":
35  43        main()
36      -
37      -
```

108 utils/futter_loader.py

```
...  ...   @@ -1,39 +1,81 @@
1        - import os
     1    + from typing import List
     2    + from pathlib import Path
2    3     import csv
     4    + import logging
     5    + from models.pferd import Pferd
3    6     from models.futter import Heu, Heulage
     7    + from utils.validation import validate_pferd, validate_heu,
           validate_heulage
4    8
5        - def lade_heu_aus_csv(pfad: str) -> Heu:
6        -     with open(pfad, newline='', encoding='utf-8') as csvfile:
     9    + logger = logging.getLogger(__name__)
     10   +
     11   + DATA_DIR = (Path(__file__).parent.parent / "data").resolve()
     12   +
     13   + def lade_pferde_als_dataclasses(dateiname: str) -> List[Pferd]:
     14   +     pfad = DATA_DIR / dateiname
     15   +     pferde_liste = []
     16   +     with pfad.open(newline='', encoding='utf-8') as csvfile:
7    17            reader = csv.DictReader(csvfile)
8        -         row = next(reader)  # Nur die erste Zeile, da Analysewerte
           meist 1 Zeile
9        -         return Heu(
10       -             name=os.path.splitext(os.path.basename(pfad))[0],
11       -             trockenmasse=float(row['Trockensubstanz']),
12       -             rohprotein=float(row['Rohprotein']),
13       -             rohfaser=float(row['Rohfaser']),
14       -             gesamtzucker=float(row['Gesamtzucker']),
15       -             fruktan=float(row['Fruktan']),
16       -             me_pferd=float(row['ME-Pferd']),
17       -             pcv_xp=float(row.get('pcv_XP', 0)),
18       -             herkunft=None,
19       -             jahrgang=int(row.get('Jahrgang', 2025))
20       -         )
     18   +         for row in reader:
     19   +             if validate_pferd(row):
     20   +                 pferd = Pferd(
     21   +                     name=row['Name'],
     22   +                     gewicht=float(row['Gewicht']),
     23   +                     alter=int(row['Alter']),
```

```
+                    notizen=row.get('Notizen')
+                )
+                pferde_liste.append(pferd)
+            else:
+                logger.warning(f"Ungültige Pferdedaten übersprungen: {row}")
+    return pferde_liste

- def lade_heulage_aus_csv(pfad: str) -> Heulage:
-     with open(pfad, newline='', encoding='utf-8') as csvfile:
+ def lade_heu_als_dataclasses(dateiname: str) -> List[Heu]:
+     pfad = DATA_DIR / dateiname
+     heu_liste = []
+     with pfad.open(newline='', encoding='utf-8') as csvfile:
          reader = csv.DictReader(csvfile)
-         row = next(reader)
-         return Heulage(
-             name=os.path.splitext(os.path.basename(pfad))[0],
-             trockenmasse=float(row['Trockensubstanz']),
-             rohprotein=float(row['Rohprotein']),
-             rohfaser=float(row['Rohfaser']),
-             gesamtzucker=float(row['Gesamtzucker']),
-             fruktan=float(row['Fruktan']),
-             me_pferd=float(row['ME-Pferd']),
-             pcv_xp=0,
-             herkunft=None,
-             jahrgang=int(row.get('Jahrgang', 2025)),
-             ph_wert=float(row.get('pH-Wert', 0)),
-             siliergrad=None
-         )
+         for row in reader:
+             if validate_heu(row):
+                 heu = Heu(
+                     name=pfad.stem,
+                     trockenmasse=float(row['Trockensubstanz']),
+                     rohprotein=float(row['Rohprotein']),
+                     rohfaser=float(row['Rohfaser']),
+                     gesamtzucker=float(row['Gesamtzucker']),
+                     fruktan=float(row['Fruktan']),
+                     me_pferd=float(row['ME-Pferd']),
+                     pcv_xp=float(row.get('pcv_XP', 0)),
+                     herkunft=row.get('Herkunft'),
+                     jahrgang=int(row.get('Jahrgang', 2025)),
+                     staubarm=None  # Optional, falls vorhanden
+                 )
+                 heu_liste.append(heu)
+             else:
+                 logger.warning(f"Ungültige Heudaten übersprungen: {row}")
+     return heu_liste
+
```

```
+ def lade_heulage_als_dataclasses(dateiname: str) -> List[Heulage]:
+     pfad = DATA_DIR / dateiname
+     heulage_liste = []
+     with pfad.open(newline='', encoding='utf-8') as csvfile:
+         reader = csv.DictReader(csvfile)
+         for row in reader:
+             if validate_heulage(row):
+                 heulage = Heulage(
+                     name=pfad.stem,
+                     trockenmasse=float(row['Trockensubstanz']),
+                     rohprotein=float(row['Rohprotein']),
+                     rohfaser=float(row['Rohfaser']),
+                     gesamtzucker=float(row['Gesamtzucker']),
+                     fruktan=float(row['Fruktan']),
+                     me_pferd=float(row['ME-Pferd']),
+                     pcv_xp=float(row.get('pcv_XP', 0)),
+                     herkunft=row.get('Herkunft'),
+                     jahrgang=int(row.get('Jahrgang', 2025)),
+                     ph_wert=float(row.get('pH-Wert', 0)),
+                     siliergrad=row.get('Siliergrad')
+                 )
+                 heulage_liste.append(heulage)
+             else:
+                 logger.warning(f"Ungültige Heulagedaten übersprungen:
  {row}")
```

```
80 +     return heulage_liste
81 +
```

35 utils/validation.py

```
...  ...  @@ -1,4 +1,35 @@
 1    1   # utils/validation.py
      2  + import logging
      3  +
      4  + logger = logging.getLogger(__name__)
      5  +
 2    6   def validate_pferd(pferd: dict) -> bool:
 3        -     required = ['Name', 'Gewicht', 'Alter']
 4        -     return all(f in pferd and pferd[f] for f in required)
      7  +     """Validiert die wichtigsten Felder eines Pferds."""
      8  +     required_fields = ['Name', 'Gewicht', 'Alter']
      9  +     for field in required_fields:
     10  +         if field not in pferd:
     11  +             logger.error(f"Fehlendes Feld in Pferdedaten: {field}")
     12  +             return False
     13  +     if not isinstance(pferd['Gewicht'], (int, float)) or
          pferd['Gewicht'] <= 0:
     14  +         logger.error(f"Ungültiges Gewicht: {pferd['Gewicht']}")
     15  +         return False
     16  +     if not isinstance(pferd['Alter'], int) or pferd['Alter'] <= 0:
     17  +         logger.error(f"Ungültiges Alter: {pferd['Alter']}")
     18  +         return False
     19  +     return True
     20  +
     21  + def validate_heu(heu: dict) -> bool:
     22  +     required_fields = ['Trockensubstanz', 'Rohprotein', 'Rohfaser',
          'Gesamtzucker', 'Fruktan', 'ME-Pferd']
     23  +     for field in required_fields:
     24  +         if field not in heu or heu[field] == "":
     25  +             logger.error(f"Fehlendes Feld in Heudaten: {field}")
     26  +             return False
     27  +     return True
     28  +
     29  + def validate_heulage(heulage: dict) -> bool:
     30  +     required_fields = ['Trockensubstanz', 'Rohprotein', 'Rohfaser',
          'Gesamtzucker', 'Fruktan', 'ME-Pferd']
     31  +     for field in required_fields:
     32  +         if field not in heulage or heulage[field] == "":
     33  +             logger.error(f"Fehlendes Feld in Heulagedaten: {field}")
     34  +             return False
     35  +     return True
```

100 views/main_window.py

```
...  ...  @@ -1,52 +1,110 @@
 1    1   # views/main_window.py
 2    2
 3        - from PyQt5.QtWidgets import QMainWindow, QLabel, QPushButton,
           QVBoxLayout, QWidget, QComboBox
      3  + from PyQt5.QtWidgets import QMainWindow, QWidget, QVBoxLayout,
           QLabel, QPushButton, QComboBox, QStackedWidget
 4    4   from PyQt5.QtCore import QTimer
      5  + import os
      6  + from utils.futter_loader import lade_heu_aus_csv
      7  + from views.start import StartSeite
 5    8
 6    9   class MainWindow(QMainWindow):
 7   10       def __init__(self, sensor, heu_namen=None):
 8   11           super().__init__()
 9   12           self.sensor = sensor
     13  +         self.status = "start"
     14  +         self.heu_namen = heu_namen if heu_namen is not None else []
     15  +         self.init_ui()
10   16
     17  +     def init_ui(self):
11   18           self.setWindowTitle("Futterkarre 2.0")
12   19           self.setFixedSize(1024, 600)
13   20
14        -          # Gewichtsanzeige
15        -          self.weight_label = QLabel("Gewicht: -- kg")
16        -          self.refresh_button = QPushButton("Aktualisieren")
     21  +          self.stacked_widget = QStackedWidget()
17   22
18        -          # Dropdown für Heu-Auswahl
19        -          self.combo_heu = QComboBox()
20        -          if heu_namen is None:
21        -              heu_namen = ["heu2024", "heu2025", "heu_nachbar2025"]
22        -          self.combo_heu.addItems(heu_namen)
23        -
           self.combo_heu.currentIndexChanged.connect(self.on_heu_changed)
     23  +          # Startbildschirm aus start.ui
     24  +          self.start_screen = StartSeite()
     25  +          self.load_screen = self.create_load_screen()
     26  +          self.feed_screen = self.create_feed_screen()
     27  +          self.summary_screen = self.create_summary_screen()
     28  +
     29  +          self.stacked_widget.addWidget(self.start_screen)   # Index 0
     30  +          self.stacked_widget.addWidget(self.load_screen)    # Index 1
     31  +          self.stacked_widget.addWidget(self.feed_screen)    # Index 2
     32  +          self.stacked_widget.addWidget(self.summary_screen) # Index 3
     33  +
     34  +          self.setCentralWidget(self.stacked_widget)
     35  +          self.show_status("start")
24   36
```

```
37 +         # Button-Event verbinden: Nach Klick auf START geht es zum
   Beladen
38 +         self.start_screen.btn_start.clicked.connect(lambda:
   self.show_status("beladen"))
39 +
40 +     def show_status(self, status):
41 +         self.status = status
42 +         if status == "start":
43 +             self.stacked_widget.setCurrentWidget(self.start_screen)
44 +         elif status == "beladen":
45 +             self.stacked_widget.setCurrentWidget(self.load_screen)
```

```
+            elif status == "fuettern":
+                self.stacked_widget.setCurrentWidget(self.feed_screen)
+            elif status == "abschluss":
+                self.stacked_widget.setCurrentWidget(self.summary_screen)
+
+        def create_load_screen(self):
+            widget = QWidget()
             layout = QVBoxLayout()
+            self.weight_label = QLabel("Gewicht: -- kg")
+            self.combo_heu = QComboBox()
+            self.combo_heu.addItems(self.heu_namen)
+
self.combo_heu.currentIndexChanged.connect(self.on_heu_changed)
+            self.refresh_button = QPushButton("Aktualisieren")
+            self.refresh_button.clicked.connect(self.update_weight)
+            button = QPushButton("Beladung abgeschlossen")
+            button.clicked.connect(lambda: self.show_status("fuettern"))
+            layout.addWidget(QLabel("Bitte Futter wählen und Wagen
beladen."))
             layout.addWidget(self.weight_label)
             layout.addWidget(self.combo_heu)
             layout.addWidget(self.refresh_button)
-
-            container = QWidget()
-            container.setLayout(layout)
-            self.setCentralWidget(container)
-
-            self.refresh_button.clicked.connect(self.update_weight)
-
-            # Automatische Aktualisierung alle 1 Sekunde
+            layout.addWidget(button)
+            widget.setLayout(layout)
+            # Gewicht automatisch aktualisieren
             self.timer = QTimer()
             self.timer.timeout.connect(self.update_weight)
             self.timer.start(1000)
+            return widget
+
+        def create_feed_screen(self):
+            widget = QWidget()
+            layout = QVBoxLayout()
+            label = QLabel("Fütterung läuft...")
+            button_fuettern = QPushButton("Füttern")
+            button_fuettern.clicked.connect(lambda:
self.show_status("abschluss"))
+            button_nachladen = QPushButton("Nachladen")
+            button_nachladen.clicked.connect(lambda:
self.show_status("beladen"))
+            layout.addWidget(label)
+            layout.addWidget(button_fuettern)
+            layout.addWidget(button_nachladen)
```

```
+         widget.setLayout(layout)
+         return widget
+
+     def create_summary_screen(self):
+         widget = QWidget()
+         layout = QVBoxLayout()
+         label = QLabel("Fütterung abgeschlossen!\nHier kommt die
  Checkliste.")
+         layout.addWidget(label)
+         widget.setLayout(layout)
+         return widget

     def update_weight(self):
         try:
             weight = self.sensor.read_weight()
             self.weight_label.setText(f"Gewicht: {weight:.2f} kg")
         except Exception as e:
             self.weight_label.setText("Fehler beim Wiegen!")
-            # Hier könntest du ein Logging-Framework nutzen

     def on_heu_changed(self, index):
         heu_name = self.combo_heu.currentText()
-        # Hier kannst du das neue Heu laden und ggf. anzeigen
-        print(f"Aktuell ausgewähltes Heu: {heu_name}")
+        pfad = os.path.join("data", heu_name)
+        try:
+            heu = lade_heu_aus_csv(pfad)
+            print(f"Geladenes Heu: {heu}")
+        except Exception as e:
+            print(f"Fehler beim Laden von {heu_name}: {e}")
```

12 views/start.py

```
@@ -0,0 +1,12 @@
+ import os
+ from PyQt5 import uic
+ from PyQt5.QtWidgets import QWidget
+
+ class StartSeite(QWidget):
+     def __init__(self, parent=None):
+         super().__init__(parent)
+         # Ermittle den absoluten Pfad zur start.ui relativ zu diesem
  Skript
+         current_dir = os.path.dirname(os.path.abspath(__file__))
+         ui_path = os.path.join(current_dir, 'start.ui')
+         uic.loadUi(ui_path, self)
+
```

124 views/start.ui

```
@@ -0,0 +1,124 @@
+ <?xml version="1.0" encoding="UTF-8"?>
+ <ui version="4.0">
+  <class>StartSeite</class>
+  <widget class="QWidget" name="StartSeite">
+   <property name="geometry">
+    <rect>
+     <x>0</x>
+     <y>0</y>
+     <width>1024</width>
+     <height>601</height>
+    </rect>
+   </property>
+   <property name="windowTitle">
+    <string>Futterkarre 1.0</string>
+   </property>
+   <property name="styleSheet">
+    <string notr="true">background-color: #fdffe0;</string>
+   </property>
+   <widget class="QLabel" name="label_title">
+    <property name="geometry">
+     <rect>
+      <x>0</x>
+      <y>150</y>
+      <width>1024</width>
+      <height>80</height>
+     </rect>
+    </property>
+    <property name="sizePolicy">
+     <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
+      <horstretch>0</horstretch>
+      <verstretch>0</verstretch>
+     </sizepolicy>
+    </property>
+    <property name="minimumSize">
+     <size>
+      <width>1024</width>
+      <height>80</height>
+     </size>
+    </property>
+    <property name="maximumSize">
+     <size>
+      <width>1024</width>
+      <height>80</height>
+     </size>
+    </property>
+    <property name="font">
+     <font>
+      <family>Droid Sans Fallback</family>
+      <pointsize>64</pointsize>
```

```
+      <weight>75</weight>
+      <bold>true</bold>
+     </font>
+    </property>
+    <property name="styleSheet">
+     <string notr="true">color: #222;</string>
+    </property>
+    <property name="text">
+     <string>Futterkarre 2.0</string>
+    </property>
+    <property name="alignment">
+     <set>Qt::AlignCenter</set>
+    </property>
+   </widget>
+   <widget class="QPushButton" name="btn_start">
+    <property name="enabled">
+     <bool>true</bool>
+    </property>
+    <property name="geometry">
+     <rect>
+      <x>370</x>
+      <y>360</y>
+      <width>300</width>
+      <height>120</height>
+     </rect>
+    </property>
+    <property name="sizePolicy">
+     <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
```

```
+      <horstretch>0</horstretch>
+      <verstretch>0</verstretch>
+     </sizepolicy>
+    </property>
+    <property name="minimumSize">
+     <size>
+      <width>300</width>
+      <height>120</height>
+     </size>
+    </property>
+    <property name="maximumSize">
+     <size>
+      <width>300</width>
+      <height>100</height>
+     </size>
+    </property>
+    <property name="font">
+     <font>
+      <family>Droid Sans Fallback</family>
+      <pointsize>40</pointsize>
+      <weight>75</weight>
+      <bold>true</bold>
+     </font>
+    </property>
+    <property name="styleSheet">
+     <string notr="true">
+ QPushButton {
+     background-color: #b0cfa5;
+     color: #ffffff;
+     border-radius: 18px;
+     min-width: 300px;
+     min-height: 120px;
+ }
+ QPushButton:hover {
+     background-color: #8FBC8F;
+     color: #f5f5b5;
+ }
+         </string>
+    </property>
+    <property name="text">
+     <string>START</string>
+    </property>
+   </widget>
+  </widget>
+  <resources/>
+  <connections/>
+ </ui>
```