Arbeitsplan

- [x] **1. Micropython Embed Port unter Windows**
  - [x] 1.1. Code::Blocks Umgebung einrichten
  - [x] 1.2. Kompilieren und Laden einer MPY-Binär-Datei
  - [x] 1.3. Erstes Modul registrieren & Mockup erstellen
  - [x] 1.4. Erster Entwurf der Micropython-VM Schnittstellen
  - [x] 1.5. User-Interrupt Integrieren & Thread-Safety Testen/Analysieren
  - [x] 1.6. Micropython- & Program-Compile-Prozess in den Code::Blocks Build Prozess einbauen

  > Nach diesem Schritt ist das Windows Grundgerüst gebaut.

  - [x] 1.7. Vollständiges Umsetzen der ORB-Funktions-Module & Mockups
  - [x] 1.8. Python-Api Dokumentation schreiben
  - [x] 1.9. Program-Compile-Prozess in Code::Blocks Build Prozess einbauen

  > Ab diesen Punkt sollten alle Funktionen unter Windows testbar sein.

  - [x] 1.10. Erstellen der Windows-Test-Spezifikation
  - [x] 1.11. Testen der Micropython-Funktionalitäten nach Spezifikation
- [x] **2. Integration in EMBitz Projekt**
  - [x] 2.1. Ausführen der VM auf einem Microcontroller (Mit MPY-Binary in die Firmware eingebaut - d.h. als uint_8t array).

  > Hier kann das ORB-Firmware Projekt als Grundgerüst genutzt werden, ohne das Verwenden der eigentlichen ORB-Funktionen. Funktionen wie usrLed.set(1) können erst einmal als Platzhalter in die ORB-C-Interfaces eingebaut werden.

  - [x] 2.2. Übertragen und Laden des Programmes aus dem Flash-Speicher

  > Verwenden des Entwicklungs-Gerüstes (2.1.).

  - [x] 2.3. Umsetzen der tatsächlichen C-Interface-Klassen
  - [x] 2.4. Erstellen der Python-Task
  - [x] 2.5. Entwicklungsgerüst erweitern um USB-Update

  > Für ORB-Monitor-Kommunikation.

  - [x] 2.6. Testen & Dokumentieren der ORB-Python-Funktionalitäten
  - [x] 2.7. Erster Entwurf der Micropython-VM-API Dokumentation

  > Festgehalten im Sphinx-Api-Doc.
- [x] **3. Vollständige Firmware Integration**
  - [x] 3.1. Wiederherstellung der ORB-Programm-Logik
  - [x] 3.2. Konflikte zwischen Tasks identifizieren & dokumentieren

  > Hier wäre ein Beispiel, dass das UserInterface Motoren ausschaltet, wenn keine AppTask läuft.

  - [x] 3.3. Konflike lösen & dokumentieren
  - [x] 3.4. Kontrollfluss Dokumentation (aus 3.2. & 3.3.) erstellen.

  > Welcher Button startet die Python-VM, welcher AppTask, etc. dieser Schritt ist abhängig von vorher getroffenen Designentscheidungen. Dies ist identisch zu der vorherigen Firmware, es musste nichts angepasst werden - daher fällt eine ausführliche Dokumentation weg.

  - [x] 3.5. Testen & Dokumentieren der ORB-Python-Funktionalitäten im Zusammenhang mit Kontrollfluss und ORB-Firmware Kompatibilität.
  - [x] 3.6. Micropython-VM-Api Dokumentation nachbessern

- [x] 3.7. System Anforderungen an die PythonTask ermitteln
  - [x] 3.7.1. Wie groß kann / sollte die HeapSize sein?
  - [x] 3.7.1. Wie große kann / sollte der PythonTask Stack sein?
  - [x] 3.7.1. Welche Speicherblöcke bekommt die PythonTask um die MPY-Binary abzulegen?