Anforderungen

Das Ziel dieses Projektes ist die Integration eines MicroPython-Interpreters in die Firmware des Open Robotic Board (ORB). Das ORB ist eine Hardwareplattform, die den modularen Aufbau eines Roboters ermöglicht. Die ORB-Firmware stellt Funktionen, wie z.B. die Motorsteuerung, für die Roboterprogrammierung bereit. Ziel ist es, die Entwicklung von Python-Programmen für das ORB zu ermöglichen. Zusätzlich soll die Anbindung für die Python-Programmierung an den ORB-Monitor umgesetzt werden. Der ORB-Monitor ist ein Entwicklungstool, das Textausgaben, das Hochladen, Starten sowie Stoppen von MicroPython-Programmen ermöglicht.

Für Debug- und Entwicklungs-Zwecke soll auch das Compilieren und Ausführen des Micropython-Ports auf einem Entwicklungs-Rechner bereitgestellt werden.

Es folgt eine genauere Liste und Beschreibung der daraus entstehenden Anforderungen:

1. Der Micropython Embed Port kann unter Windows kompiliert und debuggt werden.
2. Die Ausführung von Micropython-Skripten soll ausschließlich durch MPY-Bytecode ermöglicht werden, nicht durch Klartext oder eine REPL-Funktion.
3. Das Python-Programm soll im Flash-Speicher des ORB abgelegt werden.
4. Das Python-Programm soll vom Nutzer gestartet und gestoppt werden können.
5. Die Übertragung des Python-Programms soll über den von DFU bereitgestellten Prozess erfolgen können. Die Firmware soll nicht erneut aufgespielt werden müssen.
6. Das Python-Programms soll über die durch den ORB-Monitor bereit gestellten Prozesse übertragen werden können. Die Firmware soll nicht erneut aufgespielt werden müssen.
7. Es sollen Micropython-Module und -Funktionen bereitgestellt werden, die sich aus den in `orblocal.h` definierten Funktionen ableiten.
   Diese Funktionen werden von der ORB-Firmware für die Programmierung in C++ zur Verfügung gestellt:

```
virtual void        configMotor      (BYTE id,WORD t,BYTE a,BYTE Kp,BYTE Ki);
virtual void        setMotor         (BYTE id,BYTE m, short s, int p       );
virtual ORB::Motor  getMotor         (BYTE id                              );
virtual void        setModelServo    (BYTE id, BYTE s, BYTE w              );
virtual DWORD       getTime          (                                     );
virtual void        wait             (DWORD time                           );
virtual void        setMonitorText   (BYTE line,const char *fmt, va_list va);
virtual BYTE        getMonitorKey    (                                     );
virtual void        clearMemory      (                                     );
virtual void        setMemory        (DWORD addr, BYTE *data, DWORD s      );
virtual void        getMemory        (DWORD addr, BYTE *data, DWORD s      );
virtual void        configSensor     (BYTE id,BYTE t,BYTE m,WORD o         );
virtual ORB::Sensor getSensor        (BYTE id                              );
virtual WORD        getSensorValueExt(BYTE id, BYTE ch                     );
virtual BYTE        getSensorDigital (BYTE id                              );
```

[vgl. ORB-FW, 'ORB-Firmware/Src/ORBlocal.h', ab Zeile 47]

8. Die ORB-Funktionen müssen in ihrer vollständigen Funktionalität, in die MP-VM, integriert werden.
   Die Verwendung von Micropython darf keinen Funktionalitätsverlust zur Folge haben.
9. Diese Funktionen sollen nicht direkt übernommen, sondern in die Python-Welt übertragen werden. Falls angebracht werden Python-Objekte/Module bereitgestellt , welche die genannten Funktionen möglichst gut und mit geeigneten Name-Spaces abbilden.
10. Auch wenn die ORB-Funktionen vollständig umgesetzt werden sollen so reicht es für Funktionen welche meherere Komponenten geleichzeitig umsetzen, diese nur für ausgewählte Komponenten zu testen. Dies würde zum Beispiel die Sensor-Klasse betreffen. Es soll lediglich sicher gestellt werden, dass alle in Micropython eingestellten Parameter korrekt an die ORB-Firmware übergeben werden und einstellbar sind. Ist dies gegeben, so gilt eine Funktion als getestet.

11. Es sollen keine standardmäßigen Micropython-Module ge-ported werden. Es werden ausschließlich die Funktionen der ORB-Firmware bereitgestellt, sowie ausgewählte Hilfsfunktionen wie z.B. `math`, `min`, `max` usw..
12. Es soll eine klar definierte Schnittstelle zur Python-VM existieren, die als einzige Verbindung zwischen Micropython und der ORB-Firmware dient.
13. Es soll eine Python-Task implementiert werden, die parallel zu den anderen Aufgaben der ORB-Firmware läuft. Diese Python-Task nutzt die Schnittstelle zur Python-VM.
14. Die Funktionen der ORB-Firmware sollen lediglich erweitert und nicht grundlegend verändert werden.
    Die Python-Task muss im Einklang mit den restlichen Aufgaben der ORB-Firmware ausgeführt werden können.
    Anpassungen wie z.B. die Neubelegung von Tasten oder das Verschieben sowie Verringern von Nutzer-Speicherblöcken sind erlaubt, solange diese die Funktionalität der ORB-Firmware nicht einschränken.
15. Den Nutzern sollen ausreichende Informationen zur Verfügung gestellt werden, damit sie diese Funktionen verwenden können. Das heißt, auch ohne Vorkenntnisse über die ORB-Firmware soll es möglich sein, den Micropython-Interpreter zu nutzen. Dies sollte in der Form eines API-Dokumentes gestaltet werden.