Turbo SDK - MC Integration options

No integration

Planning based on reducing SDK dependencies in order to reach the End of March internal release of MC.

PROs	CONs
Minimum regression risk	Forced SDK API freeze
Reach E-o-March deadline	Double implementation
No SDK depedencies	

Task List

SPs	Description
5	Refactor direct use of low energy property provider
5	extending LowEnergyPropertyProvider • low protocol implementation (BLE event handlers, protocol spec)
13	BikeData (high level protocol) • make sure that we are reading properties before writing (caching value • new implementation for this protocol
8	Extend BikeData factory logic within LowEnergyConnector

8	Device Scanner Filter (bike identification on scan)
?	ConnectableDeviceStore (known bikes, including device type)- • Q: how can we identify a known bike which protocol is using without s
sp.	Total
39	approx. **6 weeks**

MC Integrates SDK Core component

SDK Requirements:

- Read/Write properties (in progress)
- New TCU 2.0 protocol
  - new bike identification (BLE DIS)
  - protocol identification (on connect)
  - implementation
- Multi-packet/page support (in progress)

PROs	**CONs**
Reuse protocol impl.	SDK timeline dependencies
No native layer (JNI)	Cannot reach E-o-March deadline
–	Only a small part of SDK is used
–	Does not reduce MC complexity
–	High regression risk

SPs	**Task list**
5 sp	Core library setup (build with QMake)
5 sp	Refactor direct use of low energy property provider

8 sp	Identification sequence (allows protocol instantiation + DIS)
5 sp	Notify request handling -> LowEnergyPropertyProvider
13 sp	Read request sequence impl. -> LowEnergyPropertyProvider
8 sp	Write request sequence impl. -> LowEnergyPropertyProvider
–	Total
44 sp	6 weeks

MC - Full native SDK integration

PROs	CONs
Full use of SDK features	High regression risk
Reduces MC complexity	SDK timeline dependencies
Internal SDK validation	Cannot reach E-o-March deadline

SDK New protocol estimation

SPs	Description
5	Core identification sequence
8	Enhance iprotocol with list of parameters (manual)
8	Core protocol parsing • Code generation and templates
8 x 2	Jni, internal API changes for list of params, forward change notifications
8	Core properties cache - cache impl. in core - diff and change notificatior
6 + 3	Notify requests handling (Native + Core)
6 + 5	Read requests sequence (Native + Core)

10 + 5	Write requests sequence (Native + Core)
3 sp	Extend Android api for using an external notification object - MC modif
3 sp	Ensure all iOS apis are exposed as objc (Ensure the new read-write can l
3 sp	Ensure compatibility with crashlytics (expose debug symbols)
–	Total
42 sp	4 weeks (core only)
80 sp	6-8 weeks (native + core)

Mission Control Estimation

SDK required features:

- Bluetooth communication: scanning, connection, communication
- Write single packet property
- Read & Write Multi-packet and Multi-page properties
- New TCU 2.0 protocol support

Assumptions:

- Error library can work in its if has access to ERROR_LOG property.
- Mission control will handle network connections, authentication, persistence, ride info, presets.

? HeartRate needs a different BLE connection for heart rate sensor, for another device ?

SPs	Task list:
8 sp	native libraries setup
5 sp	Refactor direct use of low energy property provider
13 sp	Replace BLE scanning modules - JNI layer for all api calls - iOS layer ...
8 sp	ConnectionManager updates
13 sp	Wrapping SDK encode/decode feature into LowEnergyPropertyProvider

8 sp	testing BikeData with fake/mock property provider
–	Total
55 sp	6-8 weeks

Definition: This setup will allow mission control to work with the new bike (reuse protocols from SDK) and maintain current functionality.