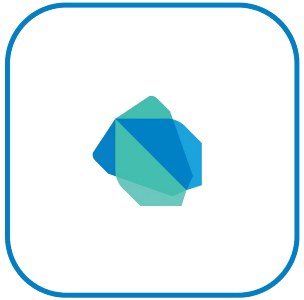


Dartino

Dartino: Modern Embedded Programming

Product summary, February 2016

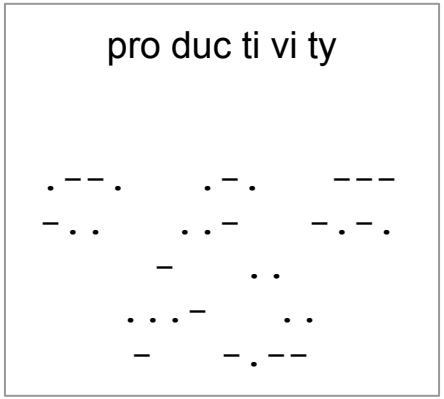
Google

Introducing Dartino

- Experiencing vast growth in embedded devices & IoT?
- Need to quickly prototype, customize, and productize?
- Creating graphical user interfaces and/or secure communication over BLE and WiFi?

Writing application code using C/C++ is costly and error prone.

Dartino enables a modern programming language on embedded microcontrollers, reducing development cost by saving time and improving quality and security.



Modern Programming for Embedded

- Program embedded with a modern, high-level language, *Dart*
- Elegant and familiar syntax
- Easy to learn based on current experience from C, C++, Java, JavaScript, Python, C#, etc.
- Optional typing: Agile prototyping yet strong validation
- Better testing: Easy to run and automate code on a PC

```
main() {
  // Turn on the status LED
  const int statusLed = 4;
  var gpio = new GPIO();
  gpio.setMode(statusLed, Mode.output);
  gpio.setPin(statusLed, true);

  // Send current temperature as a MQTT message
  Client c = new Client(
    'tcp://test.mosquitto.org:1883', 'test-client');
  c.publish('Temp: $currentTemp', '/sensors/temp');
  c.disconnect();
}
```

Interoperability with existing C/C++ and native code

- Combine Dartino code with existing code enabling reuse
- Generate Dartino wrapper libraries automatically from header files [IN PROGRESS]

```
// Initialize a library from a shared library file (.so/.dll)
final ForeignLibrary mqttLib = new ForeignLibrary.fromName(
    ForeignLibrary.bundleLibraryName('paho-mqtt3c'));

// Call function in native code and return result to Dartino code
ForeignMemory mqttClientHandle = new ForeignMemory.allocatedFinalized(8);
int result = mqttLib.lookup('MQTTClient_create').icall$5(
    mqttClientHandle,
    new ForeignMemory.fromStringAsUTF8(serverURI),
    new ForeignMemory.fromStringAsUTF8(clientID),
    1,
    ForeignPointer.NULL);
```

Free & Open Source

- Dartino is free to use. This includes both developer tools & the runtime

- Dartino is developed as a full open source project on GitHub
  - Transparent development process
  - Community participation encouraged
  - Endless customization opportunities

- Integrates with other open source
  - Embedded OS: FreeRTOS link
  - Sockets & TCP/IP: FreeRTOS+TCP link
  - SSL/TLS: mbed TLS link

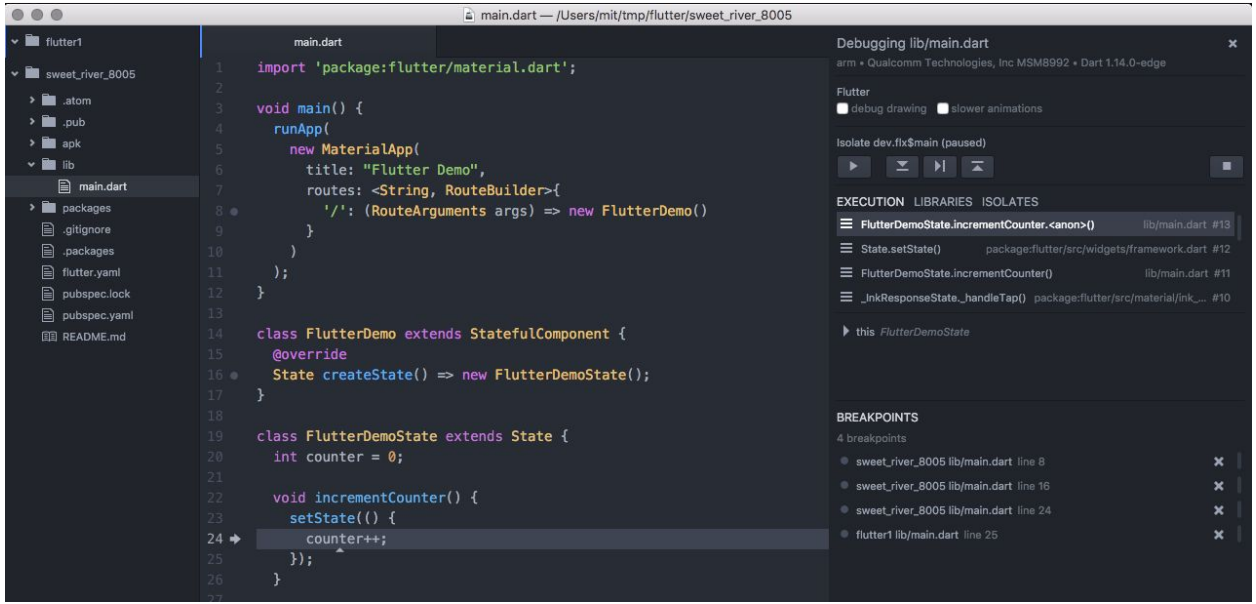
https://github.com/dartino/

Hardware platform support

- Dartino code can be run both locally on the developer PC, and on a attached embedded device
- Runtime is available on ARM Cortex processors
  - Medium to large micro-controllers (MCUs):
    - Cortex M3, M4, and M7 processors
    - 512kb flash or more
    - 128kb RAM or more
  - Micro-processors (MPUs)
    - Raspberry Pi 2
- Integrates with developer and custom boards using FreeRTOS and silicon vendor drivers and HALs

Tools

- Free open source tools
  - Command line compilation, image building, and flashing
  - IDE with syntax highlighting, go to declaration, etc.
  - Debugger (breakpoints, stack, view variables, etc.)
    [IN PROGRESS]



Additional info and contact details

Email (Dartino Product Manager):
mit@google.com

Homepage [IN PROGRESS]:
https://dartino.org

GitHub repo:
https://github.com/dartino/sdk