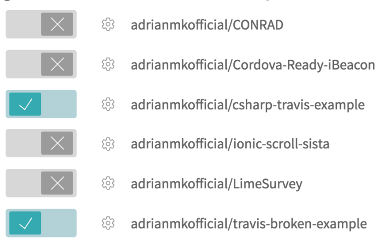
TRAVIS CI

Travis CI[1] (as required by the “AMOS” Module) in its simplest form is basically a free to use online service that runs tasks after a PUSH or MERGE occurs on a GitHub Repository. This ensures that each commit made to the repo by developers doesn’t break the tests. For this it uses the “`.travis.yml`“ file, a configuration file that tells the automated task runner to build the project and then run tests or other scripts (e.g. shell scripts).

Travis offers some great functionality to ease continuous integration of all code uploaded to the GitHub repo, e.g. by offering notification of build status via mail, automatically running custom bash scripts, installing dependencies, building the project, and running unit tests.



✗	adrianmkofficial/CONRAD
✗	adrianmkofficial/Cordova-Ready-iBeacon
✓	adrianmkofficial/csharp-travis-example
✗	adrianmkofficial/ionic-scroll-sista
✗	adrianmkofficial/LimeSurvey
✓	adrianmkofficial/travis-broken-example

Travis allows to easily synchronize projects on GitHub just by enabling a button. After this, builds can be configured, and run. The current status of a build can be reviewed on the web interface.

The builds and tests are executed in isolated virtual containers on systems running either Linux or OSX. C# as a programming language is only supported on a community-basis and uses either (by default) the Mono Framework or .NET Core libraries, so there’s a possibility one might run into problems due to unsupported Windows .NET framework, libraries and also UWP applications. The following example shows how a generic project might be build and tested with the help of the testing framework xUnit[2].

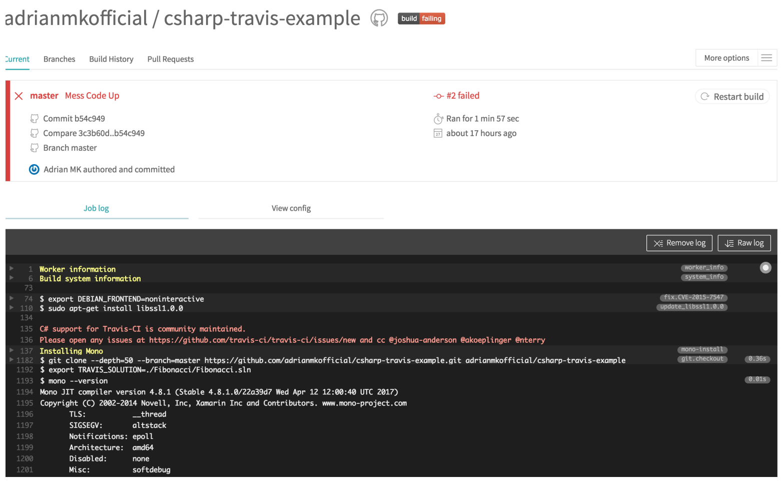
```
language: csharp
solution: "path_to_solution.sln"
matrix:
     include:
     - dotnet: 1.0.1
       mono: none
      - mono: latest
install:
- travis_retry nuget restore -source "https://www.nuget.org/api/v2"
./path/solution.sln
script:
- xbuild ./path/solution.sln
- mono ./path/packages/xunit.runner.console*/tools/xunit.console.exe
./path/solution.Tests/bin/Debug/solution.Tests.dll
```

Example Travis.yml-File

After initial trials (setting up test project, following tutorial, adjusting to C# environment) of the system Travis CI, while easily useable and configurable, proved to be unsuited for the development on a Raspberry Pi and the Microsoft-based technology stack. This is because it doesn’t support UWP-based applications right out-of-the-box. Future development might also cause issues with this testing environment because of incompatible libraries that don’t cross-compile for Mono or .NET Core.

[1] https://travis-ci.org/
[2] https://xunit.github.io/

Besides that, most of the code will depend heavily on Raspberry Pi-specific functionality (use of GPIO), which is not automatically and easily testable in the cloud[3].



master Mess Code Up
#2 failed
Commit b54c949
Compare 3c3b60d..b54c949
Branch master
Adrian MK authored and committed
Ran for 1 min 57 sec
about 17 hours ago

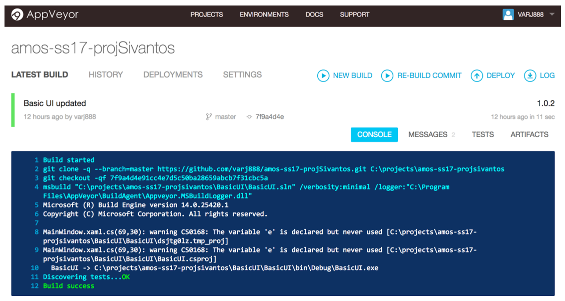
```
Worker information                                                                    worker_info
Build system information                                                              system_info

$ export DEBIAN_FRONTEND=noninteractive                                               fix.CVE-2015-7547
$ sudo apt-get install libssl1.0.0                                                    update_libssl1.0.0

C# support for Travis-CI is community maintained.
Please open any issues at https://github.com/travis-ci/travis-ci/issues/new and cc @joshua-anderson @akoeplinger @nterry
Installing Mono                                                                       mono-install
$ git clone --depth=50 --branch=master https://github.com/adrianmkofficial/csharp-travis-example.git adrianmkofficial/csharp-travis-example   git.checkout   0.36s
$ export TRAVIS_SOLUTION=./Fibonacci/Fibonacci.sln
$ mono --version                                                                      0.01s
Mono JIT compiler version 4.8.1 (Stable 4.8.1.0/22a39d7 Wed Apr 12 12:00:40 UTC 2017)
Copyright (C) 2002-2014 Novell, Inc, Xamarin Inc and Contributors. www.mono-project.com
        TLS:           __thread
        SIGSEGV:       altstack
        Notifications: epoll
        Architecture:  amd64
        Disabled:      none
        Misc:          softdebug
```

Main Interface of Travis with a failed Build

APPVEYOR

AppVeyor is also another continuous integration and deployment service like Travis CI but mainly for the Microsoft and Windows ecosystem. AppVeyor is also configured using a Web UI, or by adding a file named appveyor.yml, which is a YAML format text file, to the root directory of the code repository[4].

During the research for the possible applicability of Travis CI and AppVeyor for our project, which included reading documentation and testing both environments on sample examples (documentation and "How-To"s for beginners [5] [6] [7]) information about the testing frameworks and runtime environments which are supported in Travis CI and AppVeyor specifically for C#), we came to the conclusion, that AppVeyor is better alternative to Travis CI. It has multiple advantages (in consideration of our project settings): It is designed specially for Windows and supports full functionality of .NET Framework. It is possible to configure either with GUI or with .yml-file. It also supports numerous testing frameworks in Windows environment. It is also free for open source projects and integrated with GitHub.



amos-ss17-projSivantos

Basic UI updated
12 hours ago by varj888
master
7f9a4d4e
1.0.2
12 hours ago in 11 sec

```
Build started
git clone -q --branch=master https://github.com/varj888/amos-ss17-projSivantos.git C:\projects\amos-ss17-projsivantos
git checkout -qf 7f9a4d4e91cc4e7d5c50ba28659abcb7f31cbc5a
msbuild "C:\projects\amos-ss17-projsivantos\BasicUI\BasicUI.sln" /verbosity:minimal /logger:"C:\Program
Files\AppVeyor\BuildAgent\Appveyor.MSBuildLogger.dll"
Microsoft (R) Build Engine version 14.0.25420.1
Copyright (C) Microsoft Corporation. All rights reserved.

MainWindow.xaml.cs(69,30): warning CS0168: The variable 'e' is declared but never used [C:\projects\amos-ss17-
projsivantos\BasicUI\BasicUI\dsjtg0lz.tmp_proj]
MainWindow.xaml.cs(69,30): warning CS0168: The variable 'e' is declared but never used [C:\projects\amos-ss17-
projsivantos\BasicUI\BasicUI\BasicUI.csproj]
  BasicUI -> C:\projects\amos-ss17-projsivantos\BasicUI\BasicUI\bin\Debug\BasicUI.exe
Discovering tests...OK
Build success
```

Main Interface of AppVeyor with a succeeded build

[3] https://stackoverflow.com/questions/36805499/unit-testing-a-c-sharp-windows-10-iot-core-application-error-dep0700
[4] https://www.appveyor.com/
[5] https://docs.travis-ci.com/user/for-beginners
[6] https://docs.travis-ci.com/user/getting-started/
[7] https://www.appveyor.com/docs/

Travis CI (as described above) when compared to AppVeyor offers limited functionality for testing and deployment specifically for C#-Projects and will probably lead to further complications in the development process.

VERDICT

In the direct comparison of Travis CI and AppVeyor, the later one proves to be the better option due to its support of a Microsoft Windows based technology stack. This is because the very essence of Travis CI is build and perfectly suited rather for a Java or Web-based technology stack and not the specific use case of this project.

Regardless of using either service there remains the issue of testing hardware-specific code directly on the Raspberry Pi.

One possible make-shift solution for this would be to hook up a Raspberry Pi to a network with a static IP, have it run constantly, and have Travis or AppVeyor build the project and afterwards run a custom shell script that basically creates a remote connection to the Raspberry Pi. Then either a shell is launched (e.g. via ssh) on the actual device, code is pulled via the GitHub repository or somehow build via Travis or AppVeyor and deployed. The code needs then to execute and the results are returned back to the CI VM that's running. Also refer to Custom Deployment on Travis CI for this[8], or AppVeyor Deployment options.[9]

A much simpler alternative without the need for a lot of set-up is conducting manual testing by using Remote Debugging Tools for Visual Studio, but without the ease of automation.

[8] https://docs.travis-ci.com/user/deployment/custom/
[9] https://www.appveyor.com/docs/deployment/ftp/