

UCLA Robots Lab 2014

Generated by Doxygen 1.8.7

Thu Aug 14 2014 18:45:41

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	File Index	3
2.1	File List	3
3	Namespace Documentation	5
3.1	databaseManager Namespace Reference	5
3.1.1	Detailed Description	5
3.1.2	Function Documentation	6
3.1.2.1	clearDB	6
3.1.2.2	connectDB	6
3.1.2.3	createDB	6
3.1.2.4	deleteDB	6
3.1.2.5	htmlBody	6
3.1.2.6	htmlTail	6
3.1.2.7	htmlTop	6
3.2	robotServer Namespace Reference	6
3.2.1	Detailed Description	8
3.2.2	Function Documentation	8
3.2.2.1	connectDB	8
3.2.2.2	crossProduct	8
3.2.2.3	dotProduct	9
3.2.2.4	findCurrentEntry	9
3.2.2.5	findMaxTime	9
3.2.2.6	findNextTime	9
3.2.2.7	genNewDests	9
3.2.2.8	getDataCollected	10
3.2.2.9	getNextDest	10
3.2.2.10	htmlForm	10
3.2.2.11	htmlResponse	10
3.2.2.12	inNewLocation	10

3.2.2.13	jsonResponse	11
3.2.2.14	locate	11
3.2.2.15	numDataCollected	11
3.2.2.16	saveEndDataToDB	11
3.2.2.17	saveStartToDB	11
3.2.2.18	saveStateToDB	12
3.2.2.19	updateNextPaths	12
3.3	vehicle_tracker Namespace Reference	12
3.3.1	Detailed Description	13
3.3.2	Function Documentation	13
3.3.2.1	camProcessing	13
3.3.2.2	findYOriginShift	13
3.3.2.3	getHeading	14
3.3.2.4	headerFitsInTag	14
3.3.2.5	sendData	14
4	File Documentation	15
4.1	compressedSensing/compressedSensing.ino File Reference	15
4.1.1	Detailed Description	16
4.1.2	Function Documentation	17
4.1.2.1	displayConnectionDetails	17
4.1.2.2	get_response	17
4.1.2.3	loop	17
4.1.2.4	send_request	17
4.1.2.5	setup	17
4.1.2.6	setupConnection	18
4.1.3	Variable Documentation	18
4.1.3.1	cc3000	18

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

databaseManager	A support script for robotServer.cgi that allows easy creation, clearing and deletion of databases	5
robotServer	Server-Side Script that handles communication with the robot. Saves the requests from the robot, the data it collects and the robot's position (as tracked by the overhead video camera), to a MySQL DB. A MATLAB script directs the robot where to go, using adaptive path selection . .	6
vehicle_tracker	Python Script that uses the overhead video camera to track a single robot with a black-and-white tag on the test bed	12

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

compressedSensing/ compressedSensing.ino	
Arduino code to collect and communicate data to a server which can reconstruct it using a	
compressed sensing algorithm	15

Chapter 3

Namespace Documentation

3.1 databaseManager Namespace Reference

A support script for robotServer.cgi that allows easy creation, clearing and deletion of databases.

Functions

- def [htmlTop](#)
HTML HANDLERS.
- def [htmlBody](#)
- def [htmlTail](#)
- def [connectDB](#)
MySQL HANDLERS.
- def [createDB](#)
- def [clearDB](#)
- def [deleteDB](#)

Variables

- list [tables](#) = ["State_Record", "Data_Collection", "Next_Paths"]
MAIN PROGRAM.
- tuple **formData** = cgi.FieldStorage()
- tuple **dbNameCreate** = formData.getvalue("dbnamecreate")
- tuple **dbNameClear** = formData.getvalue("dbnameclear")
- tuple **dbNameDelete** = formData.getvalue("dbnamedelete")

3.1.1 Detailed Description

A support script for robotServer.cgi that allows easy creation, clearing and deletion of databases.

Author

Siddarth Srinivasan (UCLA REU 2014)

Date

8th July 2014

3.1.2 Function Documentation

3.1.2.1 `def databaseManager.clearDB (db, cursor, dbname, tables)`

`@brief` Function that clears the database with name 'dbname' of all 'tables'

3.1.2.2 `def databaseManager.connectDB ()`

MySQL HANDLERS.

`@brief` Function that connects to the database

3.1.2.3 `def databaseManager.createDB (db, cursor, dbname)`

`@brief` Function that creates a database with name 'dbname'

3.1.2.4 `def databaseManager.deleteDB (db, cursor, dbname)`

`@brief` Function that deletes the database with name 'dbname'

3.1.2.5 `def databaseManager.htmlBody ()`

`@brief` Function that generates the body of the page

3.1.2.6 `def databaseManager.htmlTail ()`

`@brief` Function that generates the closing html tags

3.1.2.7 `def databaseManager.htmlTop ()`

HTML HANDLERS.

`@brief` Function that generates the html tags till the body.

3.2 robotServer Namespace Reference

Server-Side Script that handles communication with the robot. Saves the requests from the robot, the data it collects and the robot's position (as tracked by the overhead video camera), to a MySQL DB. A MATLAB script directs the robot where to go, using adaptive path selection.

Functions

- `def` [htmlForm](#)
HTML HANDLERS.
- `def` [htmlResponse](#)
- `def` [jsonResponse](#)

- def [locate](#)
CAMERA/LOCATION HANDLERS.
- def [findMaxTime](#)
- def [findNextTime](#)
- def [dotProduct](#)
- def [crossProduct](#)
- def [connectDB](#)
MySQL DATABASE HANDLERS.
- def [saveStateToDB](#)
- def [saveStartToDB](#)
- def [saveEndDataToDB](#)
- def [numDataCollected](#)
- def [findCurrentEntry](#)
- def [inNewLocation](#)
- def [getNextDest](#)
MATLAB DATABASE HANDLERS.
- def [genNewDests](#)
- def [updateNextPaths](#)
- def [getDataCollected](#)

Variables

- int **X_MIN** = 40
- int **Y_MIN** = 80
- int **X_MAX** = 600
- int **Y_MAX** = 800
- int **NO_ERROR_M** = -1
- int **NO_ERROR_T** = 0
- int **NNL_ERROR** = 1
- int **S_TIMEOUT_ERROR** = 2
- string **COM_PORT** = 'COM6'
- int **BAUD_RATE** = 115200
- int **SER_TIMEOUT** = 2
- int **TIMEOUT** = 3
- int **NUM_SERIAL_DATA** = 4
- int **RADIUS_CUTOFF** = 1
- int **MIN_TIME_TO_TRAVEL** = 1500
- int **NUM_PATHS_TO_ADD** = 20
- string **DIM** = "[900 700]"
- string **SCALE** = "0.1"
- **USE_ADAPTIVE** = True
- string **BOUNDS** = "[100 120 540 760]"
- float **PIXELS_PER_SECOND** = 170.0
- float **RADS_PER_SECOND** = 1.30
- string **dbName** = "Log7"
- int **NO_DATA** = -1
- tuple **numDataPt** = [numDataCollected](#)(dbName)
MAIN PROGRAM.
- tuple **currDataEntry** = [findCurrentEntry](#)(dbName)
- tuple **submittedData** = cgi.FieldStorage()
- tuple **state** = submittedData.getvalue("state")
- **data** = NO_DATA
- **resp** = False

- int **nextTime** = 0
- **errorCode** = NO_ERROR_T
- tuple **x** = int(x)
- tuple **y** = int(y)
- tuple **theta** = float(theta)
- **respX** = x
- **respY** = y
- **respTheta** = theta

3.2.1 Detailed Description

Server-Side Script that handles communication with the robot. Saves the requests from the robot, the data it collects and the robot's position (as tracked by the overhead video camera), to a MySQL DB. A MATLAB script directs the robot where to go, using adaptive path selection.

Author

Siddarth Srinivasan (UCLA REU 2014)

Date

8th July 2014

Remarks

State Definitions (as submitted to the server):

State 0: Arduino is asking if the video camera has its start position. If so, the server will send a json response "True", telling the robot it can begin its path. If not, the server will send a json response "False" telling the robot that it has not been located, and so it should wait. State 1: Arduino is asking if the video camera has its end position. If so, the server will send a json response "True", telling the robot it can move to the starting point of its new path. If not, the server will send a json response "False" telling the robot that it has not been located, and so it should wait.

Error Codes: The server reports these to the database with every response

NO_ERROR_M -1 : No Error, but will travel a "maximum" distance instead of the directed distance. If this error code is obtained time sent to the robot is 0, the robot is out of bounds. NO_ERROR_T 0 : No Error, robot should travel distance to next point NNL_ERROR 1 : Not a New Location Error S_TIMEOUT_ERROR 2 : Serial Timeout Error

3.2.2 Function Documentation

3.2.2.1 def robotServer.connectDB (dbName)

MySQL DATABASE HANDLERS.

```
@brief Connects to the database with name 'dbName'

@returns db and cursor objects.
```

3.2.2.2 def robotServer.crossProduct (a, b)

```
@brief Helper function - returns cross product of two vectors a and b
```

3.2.2.3 def robotServer.dotProduct (*a*, *b*)

@brief Helper function - returns dot product of two vectors *a* and *b*

3.2.2.4 def robotServer.findCurrentEntry (*dbName*)

@brief Given the name of the database, returns the DataPtID of the most recent entry in the database.
 @param *dbName* The database from which to find current entry.
 @returns The value of DataPtID, the primary key, for the database.
 @remarks Not always the same as numDataCollected() as there may be insertions or deletions in the database.

3.2.2.5 def robotServer.findMaxTime (*x*, *y*, *theta*)

@brief Given the robot's position and heading, the function returns the max time for which the robot can travel before it will go off the edge.
 @param *x* The x coordinate of the robot
 @param *y* The y coordinate of the robot
 @param *theta* The heading of the robot
 @returns The time returned is expected to be the value used by the motors in driving the robot.

3.2.2.6 def robotServer.findNextTime (*startX*, *startY*, *theta*, *endX*, *endY*)

@brief Function that returns the duration for which the robot should move or turn in its subsequent movement to reach a given destination.
 @param *startX* The current x-coordinate of the robot
 @param *startY* The current y-coordinate of the robot
 @param *theta* The current heading of the robot
 @param *endX* The next intended x-coordinate for the robot
 @param *endY* The next intended y-coordinate for the robot
 @returns The duration the robot should move/turn for, by calculating the angle or distance to the desired end location, depending on the state of the robot.
 @remark Whether the robot turns/moves depends on the state it's in.
 @remark Also, the duration returned can be negative, which means the robot needs to turn right. A positive duration with state 1 means the robot needs to turn left.

3.2.2.7 def robotServer.genNewDests (*dbName*)

@brief Generate and save the next NUM_PATHS_TO_ADD destinations either randomly or based on previous data, by calling MATLAB's genNextTargets()
 @details This function will add the new list of destinations to existing destinations in Next_Paths. Depending on whether USE_ADAPTIVE is True or False, the call to MATLAB will either ask for adaptive paths or random paths.
 @param *dbName* The database to generate new paths in
 @remarks Uses updateNextPaths() to actually update the database and getDataCollected() to read from the database.

3.2.2.8 def robotServer.getDataCollected (dbName)

@brief Given the name of the database, returns the data collected thus far from the table Data_Collection as a string

@param dbName The database from which to pull data

@returns The startX, startY, endX, endY, Data in Data_Collection get formatted as a MATLAB matrix and returned as a string

3.2.2.9 def robotServer.getNextDest (dbName, numDataPt, state)**MATLAB DATABASE HANDLERS.**

@brief Accesses dbName to find the next destination for the robot.

@details Paths are generated NUM_PATHS_TO_ADD at a time. If all the paths in Next_Paths have been used, genNewDests() is called to generate the next set of paths.

@param dbName The db from which to get the next destination

@param numDataPt The number of data points collected so far, to correctly index the next destination

@param state New paths need only be generated once every NUM_PATHS_TO_ADD, and this happens when state = 1.

@returns The x, y coordinates of the next destination

@remarks Handles all the communication with MATLAB. None of the other functions in this section need to be called separately.

3.2.2.10 def robotServer.htmlForm ()**HTML HANDLERS.**

@brief Function that generates the html form if a state hasn't been submitted to the script.

@remarks For all practical purposes, this is only when testing so the robot should never really 'see' this code.

3.2.2.11 def robotServer.htmlResponse ()

@brief This function declares that the response will be in json if a state has been submitted to the form.

@remarks It is separate from jsonResponse() to allow print statements to be added anywhere in the code, as the response header must be declared before any response (through print statements) can be sent.

3.2.2.12 def robotServer.inNewLocation (dbName, x, y, currDataEntry)

@brief Checks if the given x, y coordinates are outside some radius of the start coordinate of the robot during that data sample, and returns true if so

@param dbName The database from which to find the last start coords

@param x The x-coord of the robot, after it has started its path

@param y The y-coord of the robot, after it has started its path

@param currDataEntry The last entry in the db, used to find start coords

@returns True if the robot's current position is outside some radius of its start position, false otherwise

3.2.2.13 def robotServer.jsonResponse (*response*)

```
@brief Returns the response in json form.

@param response    A 3-element list containing
    0) Response: True or False, depending on whether
               the video camera located the robot
    1) Duration: How long the robot should turn or
               travel along its next path.
    2) Error Code: Informs whether the locating and
               processing of the robot's position
               was successful.
```

3.2.2.14 def robotServer.locate ()**CAMERA/LOCATION HANDLERS.**

```
@brief    Function that uses the overhead video camera to locate robot

@details  The data sent by the robot-tracking script "vehicle_tracker.py"
          is read from the serial port. The data comes in the following
          format:
              *$|x|y|theta|!\n

@returns  The x, y and theta of the robot

@remarks  The origin of the grid is the bottom-left corner of the test
          bed, as seen when facing the whiteboard in MS3355.
```

3.2.2.15 def robotServer.numDataCollected (*dbName*)

```
@brief Given the name of the database, returns the number of data points
        collected in the table "Data_Collection".
@param dbName The database from which to find numDataCollected
@returns The number of entries in "Data_Collection"
```

3.2.2.16 def robotServer.saveEndDataToDB (*dbName, endX, endY, data, numDataPt*)

```
@brief    Saves the end coordinates of the robot, along with the data it
          collected to the database.
@details  Called when the video camera has located the robot and the
          robot has completed its path, function saves end coordinates to
          data table "Data_Collection".

@param dbName The database to write to
@param endX    The ending x-coord of the robot
@param endY    The ending y-coord of the robot
@param data    The data collected by the robot along this path
@param numDataPt The entry number in the database to write to
```

3.2.2.17 def robotServer.saveStartToDB (*dbName, startX, startY*)

```
@brief    Saves the starting coordinates of the robot to the database.
@details  Called when the video camera has located the robot, and the
          robot is looking to start its path. Saves info to data table
          "Data_Collection".

@param dbName The database to store to
@param startX The starting x-coord of the robot
@param startY The starting y-coord of the robot

@remarks  ASSUMES that the previous entry has been completed.
```

3.2.2.18 `def robotServer.saveStateToDB (dbName, State, Data, currentX, currentY, theta, destX, destY, Response, Duration, Error_Code)`

```
@brief    Stores the state that the robot sent, along with other useful
          information to keep track of the robot's actions.
@details  Called when the robot has contacted the server, stores info in
          the debugging table "State_Record".

@param    dbName    The database to save to
@param    State     The state the robot submitted to the server
@param    Data      The data the robot collected and submitted to server
@param    currentX  The x-coord that the video camera returned to server
@param    currentY  The y-coord that the video camera returned to server
@param    theta     The heading that the video camera returned to server
@param    destX     The x-coord the robot is trying to get to
@param    destY     The y-coord the robot is trying to get to
@param    Response  The Response that the server is sending to the robot
@param    Duration  The Duration the robot is sending to the robot
@param    Error_Code The Error_Code as reported by the server
```

3.2.2.19 `def robotServer.updateNextPaths (dbName, newPaths)`

```
@brief    Adds newPaths to Next_Paths table of database dbName.

@param    dbName    The database to update the next paths in
@param    newPaths  A string of the new paths, formatted as "x_0 y_0
                    x_1 y_1 ... x_n y_n", where x_i, y_i is the i'th
                    destination to add.
```

3.3 vehicle_tracker Namespace Reference

Python Script that uses the overhead video camera to track a single robot with a black-and-white tag on the test bed.

Functions

- `def camProcessing`
 VIDEO PROCESSING FUNCTIONS.
- `def headerFitsInTag`
- `def getHeading`
- `def camSetup`
- `def camRelease`
- `def sendData`
 MISCELLANEOUS FUNCTIONS.
- `def findYOriginShift`
- `def main`
 MAIN PROGRAM.

Variables

- `int CAM_THRESHOLD = 90`
- `int COLOUR_MAX = 255`
- `float EXPOSURE = -8.0`
- `int RESET = 500`
- `tuple PURPLE = (255, 0, 255)`
- `tuple GREEN = (0, 255, 0)`

- tuple **WHITE** = (255, 255, 255)
- int **X_OFFSET** = 12
- int **Y_OFFSET** = 412
- int **Y_ORIGIN_SHIFT** = 480
- int **THETA_SHIFT** = 2
- int **MIN_TAG_AREA** = 280
- int **MAX_TAG_AREA** = 1200
- int **MIN_HEADER_AREA** = 35
- int **MAX_HEADER_AREA** = 168
- **hasFoundRobot** = False
- string **COM_PORT** = 'COM1'
- int **BAUD_RATE** = 115200
- int **SER_TIMEOUT** = 2

3.3.1 Detailed Description

Python Script that uses the overhead video camera to track a single robot with a black-and-white tag on the test bed.

Author

Siddarth Srinivasan (UCLA REU 2014)

Date

4th August 2014

Remarks

The origin of the grid is originally at the top left corner, with x increasing across and y increasing downward, but the origin has been artificially transformed to the bottom left corner. Similarly, the positive angle was defined clockwise, but has been transformed to counter clockwise. Left and right refer to the test bed as seen when facing the whiteboard in MS3355.

3.3.2 Function Documentation

3.3.2.1 `def vehicle_tracker.camProcessing (filtered, CAM)`

VIDEO PROCESSING FUNCTIONS.

```
@brief Find contours and locate the robot
@details Takes in a thresholded image, locates the robot along with
         header strip and returns the location and heading of the robot.
@param filtered A filtered image
@param CAM      The camera number - 0 or 1 - for the two camera Setup
@returns The x-coordinate, y-coordinate and heading of the robot.
```

3.3.2.2 `def vehicle_tracker.findYOriginShift (cam)`

```
@brief Returns the y-dimension of the image, used to verify
         Y_ORIGIN_SHIFT
```

3.3.2.3 def vehicle_tracker.getHeading(tag, header)

@brief Finds the heading of the robot given the coordinates of the tag and header strip.
@details Draws a vector from the tag's center to the header's center to find the heading of the robot.

@param tag A 2-element list containing the (x,y) coordinates of the tag's center
@param header A 2-element list containing the (x,y) coordinates of the header's center

@returns The heading of the robot, in the range 0 to 2*pi.

3.3.2.4 def vehicle_tracker.headerFitsInTag(tag, header)

@brief Checks if the given header fits inside the given tag.

@param tag A 4-element list containing x, y, width, height of a tag sized contour
@param header A 4-element list containing x, y, width, height of a header sized contour

@returns True if the header fits inside the tag, False otherwise

3.3.2.5 def vehicle_tracker.sendData(xCoord, yCoord, theta)

MISCELLANEOUS FUNCTIONS.

@brief Sends the location and heading of the robot over serial

@param xCoord The x-coordinate of the robot to be sent
@param yCoord The y-coordinate of the robot to be sent
@param theta The heading of the robot to be sent

Chapter 4

File Documentation

4.1 compressedSensing/compressedSensing.ino File Reference

Arduino code to collect and communicate data to a server which can reconstruct it using a compressed sensing algorithm.

```
#include <Adafruit_CC3000.h>
#include <ccspi.h>
#include <SPI.h>
#include <string.h>
#include <stdlib.h>
#include "utility/debug.h"
#include <JsonParser.h>
```

Macros

- `#define ADAFRUIT_CC3000_IRQ 3`
- `#define ADAFRUIT_CC3000_VBAT 5`
- `#define ADAFRUIT_CC3000_CS 10`
- `#define LED 13`
- `#define LPLUS 6`
- `#define LMINUS 7`
- `#define RPLUS 8`
- `#define RMINUS 9`
- `#define IR 4`
- `#define WLAN_SSID "TP-LINK_3C1C5E"`
- `#define WLAN_PASS "20225964"`
- `#define WLAN_SECURITY WLAN_SEC_WPA2`
- `#define PREALLOC 256`
- `#define NUM_PATHS 225`
- `#define MAX_FAILS 2`
- `#define MAX_SENSOR 720`
- `#define WHITE_BOUNDARY 65`

Functions

- void `setup` (void)
The required setup function for the Arduino.
- void `initPins` ()

- Initializes the relevant pins on the arduino.*
 - void `loop` (void)
- The required loop function for the Arduino.*
 - bool `send_request` (String request)
- Sends a request to the server with the state and possibly data.*
 - void `get_response` ()
- Receives and processes the HTTP response from the server.*
 - void `forward` ()
- Motor control method to drive forward.*
 - void `backward` ()
- Motor control method to drive backward.*
 - void `rotateLeft` ()
- Motor control method to rotate left.*
 - void `rotateRight` ()
- Motor control method to rotate right.*
 - void `halt` ()
- Motor control method to stop moving.*
 - uint16_t `moveNext` (boolean state)
- Function to direct motors and sensor in collecting data.*
 - void `setupConnection` (void)
- Starts up the AdaFruit CC3000 Shield and connects to the internet.*
 - bool `displayConnectionDetails` (void)
- Retrieves the IP address and other connection details.*

Variables

- Adafruit_CC3000 **cc3000**
- Adafruit_CC3000_Client **client**
- uint32_t **ip** = cc3000.IP2U32(192,168,0,101)
- int **port** = 80
- String **repository** = "/"
- int **connectTimeout** = 15L * 10000L
- bool **state** = true
- bool **oldState** = false
- unsigned long **data** = 0
- uint16_t **paths** = 0
- long **time** = 1000
- int **numFails** = 0
- JsonParser< 8 > **parser**
- JsonHashTable **root**

4.1.1 Detailed Description

Arduino code to collect and communicate data to a server which can reconstruct it using a compressed sensing algorithm.

Authors

Siddarth Srinivasan (UCLA REU 2014)

Date

10th July 2014

4.1.2 Function Documentation

4.1.2.1 bool displayConnectionDetails (void)

Retrieves the IP address and other connection details.

Prints the IP Address, Netmask, Gateway, DHCP server and DNS server.

Remarks

The following code was taken from AdaFruit's buildTest example.

4.1.2.2 void get_response ()

Receives and processes the HTTP response from the server.

The response from the server gets stored in responseHTTP, but it also gets parsed to obtain just the json response without the headers in responseJSON. If the server's 'Response' is true, then it has identified the robot's location, so we can flip its state.

Remarks

The parsing code was taken from <http://forum.arduino.cc/index.php/topic,188902.4.0.html> It works under the assumption that there is no nesting in the json file. IT WILL NOT WORK FOR ALL JSON RESPONSES.

4.1.2.3 void loop (void)

The required loop function for the Arduino.

Handles the main logic for the Arduino. Essentially, it checks if the Arduino has travelled NUM_PATHS. If not, check how its state has changed: 1) If it has gone from state 0 to state 1, it is ready to move and collect data from the reflectance sensor. 2) If it has gone from state 1 to state 0, it is ready to move to a new starting position. 3) Otherwise, the state has not changed, so the server has not been able to successfully locate the robot, so send a request to the server again.

4.1.2.4 bool send_request (String request)

Sends a request to the server with the state and possibly data.

Connects to the server at 'ip' through 'port' using TCP, for as long as the connection does not timeout. If successfully connected, send a HTTP request to the page in the 'request' string with state and data.

Parameters

<i>request</i>	The HTTP request that is sent to the server.
----------------	--

Returns

true if the request is made successfully, false otherwise

4.1.2.5 void setup (void)

The required setup function for the Arduino.

Initializes the pins on the Arduino and sets up an internet connection.

4.1.2.6 void setupConnection (void)

Starts up the AdaFruit CC3000 Shield and connects to the internet.

Prints the amount of free RAM, checks initialization of the WiFi shield, deletes old connection profiles and obtains an IP address.

Remarks

The following code was taken from AdaFruit's buildTest example.

4.1.3 Variable Documentation

4.1.3.1 Adafruit_CC3000 cc3000

Initial value:

```
= Adafruit_CC3000(ADAFRUIT_CC3000_CS,  
                  ADAFRUIT_CC3000_IRQ,  
                  ADAFRUIT_CC3000_VBAT,  
                  SPI_CLOCK_DIVIDER)
```