Boite aux lettres connectée

Boîte aux lettre connectée à Twitter avec Arduino. Catégories Électronique, Énergie, Mobilier, Maison Difficulté Difficile Durée 10 heure(s) ① Coût 40 EUR (€) Sommaire Introduction Étape 1 - Principe de fonctionnement Étape 2 - Fabrication de la structure principale Étape 3 - Mise en place des leds Étape 4 - Câblage électronique Étape 5 - Gestion de l'alimentation Étape 6 - Utiliser Twitter avec un ESP8266 Étape 7 - Programmation de l'ESP8266 Étape 8 - Conclusion Notes et références Commentaires

Introduction

Rien de plus énervant que de devoir aller ouvrir sa boite aux lettre toutes les 5 minutes lorsque l'on attend avec impatience son nouveau PC ou le dernier composant permettant de finir un projet. Voici la solution !! Une boite aux lettres connectée vous avertissant sur Twitter lorsque le facteur passe.

C'est face à cette frustration que j'ai eu l'idée de construire une système m'avertissant quant il y a du courrier dans ma boite aux lettres.

Matériaux

Pour l'électronique:

- un ESP8266 ESP-07 et son support
- une RTC DS3231
- un régulateur 3,3V , LM1117 (ou équivalent) avec deux condensateurs de 10 µF
- un mosfet IRF520
- une diode 1N4007
- une photorésistance
- une résistance 1 KOhms et une résistance 330 Ohms
- une batterie au plomb 6V 4Ah
- un contrôleur de charge charge solaire
- deux panneaux solaires 12V 3W
- un câble U.FL vers RP-SMA de 1m environ
- une antenne wifi RP-SMA
- Ruban de leds blanches 12V
- un élévateur de tension
- un interrupteur à bascule
- des connecteurs mâles/femelles pour circuit imprimé
- un bornier
- une plaque d'essai
- du fil électrique
- de la soudure

Pour la structure:

- une planche de contreplaqué (environ 400 * 300 * 40 mm)
- 4 boulons M4 60mm
- 4 boulons M4 30mm
- de la peinture
- une boite de dérivation 80 * 80 mm
- de l'adhésif double face
- une plaque de plexiglas (environ 350 * 170 * 4 mm)
- des serflexs

Étape 1 - Principe de fonctionnement

Le courrier est détecté grâce à des LEDs et une photorésistance. Lorsque le courrier est déposé il tombe sur les LEDs disposées au fond, ce qui modifie la luminosité à l'intérieur, changement qui est détecté par la photorésistance. L'ESP 8266 détecte ainsi la présence de courrier. Il se connecte alors au réseau wifi puis publie un message sur twitter informant du passage du facteur ou du livreur. Le tout fonctionne de manière autonome grâce à des panneaux solaire et une batterie

Outils

- un fer à souder
- une scie
- tournevis
- une perceuse
- pistolet à colle chaude
- un multimètre
- une équerre
- un réglet
- matériel de protection
- convertisseur usb série



Étape 2 - Fabrication de la structure principale

Commencez par découper la plaque de contreplaqué au dimension de votre boite aux lettres.

🕅 Prenez des dimension légèrement inférieur à celle de la boite aux lettres pour pouvoir sortir et remettre facilement le système

Puis en son centre découpez un carré de 100*100 mm

Il peut être intéressant de peindre le bois pour qui résiste dans le temps

Découpez un carré de 100*100 mm dans du plexiglas.

Insérez le dans la plaque en bois. Pour le maintenir en place utilisez de la colle chaude ou de la colle plexiglas.

Enfin percez dans les coins, 4 trous de diamètre 4 puis montez y un boulon de 60mm. Ces derniers servent de pieds pour maintenir la structure en hauteur.







Étape 3 - Mise en place des leds

Pour avoir un système de LEDs suffisamment puissant pour que les variations de luminosité soient détectés je me suis inspiré des panneaux à LEDs DIY utilisés en vidéo (Cf vidéos d'Experimentboy ou DIY Perks)

Pour se faire, commencez par découper une plaque de contreplaqué de 140*140mm. Puis collez y 8/10 bandes de leds. Ensuite reliez ensemble d'un coté tous les pôles "+" et de l'autre tous les pôles "-". Et ensuite placez un bornier de l'autre coté de la plaque pour l'alimentation.

Pour finir assemblez les LEDs à la structure principale à l'aide de 4 boulons M4 30mm











Étape 4 - Câblage électronique

Comme dis précédemment le système repose sur un ESP8266 qui sera programmé à l'aide de l'IDE Arduino. Pour alimenter ce dernier il sera nécessaire d'utiliser un LM1117 couplé à deux condensateurs de 10 µF pour réguler la tension à 3,3V.

On trouve ensuite la photorésistance chargée de détecter les variations de luminosité reliée par l'intermédiaire d'une résistance de 3300hms à la broche ADC de l'ESP.

Les LEDs nécessitant un courant supérieur à ce qu'est capable de fournir l'ESP elles sont contrôlées par un l'intermédiaire d'un Mosfet (IRF520 / 530 /540) relié par l'intermédiaire d'une résistance de 1KOhms à la broche 7 (GPIO13) de l'ESP. De plus comme ces dernières doivent être alimentées en 12V et que la batterie utilisée ne fournie que 6V, il est nécessaire d'ajouter un élévateur de tension (boost converter).

Et pour finir on retrouve une RTC DS3231 qui permet au système d'être toujours à l'heure après un reboot. Cette dernière est reliée à l'ESP par l'intermédiaire d'une liaison i2C disponible sur les broches 13 (GPIO 5) et 14 (GPIO 4)

Une fois tous les composants soudés sur une plaque d'essai installez le tout dans une boite de dérivation et fixez la à la structure principale.

Si votre box internet se trouve être loin de votre boite aux lettres (comme dans mon cas) je vous conseille de déporter l'antenne wifi de l'ESP en utilisant câble U.FL vers RP-SMA et une antenne wifi RP-SMA. Vous pouvez ensuite la fixer avec les panneaux solaires par exemple.











Étape 5 - Gestion de l'alimentation

Pour alimenter le système j'ai choisi d'utiliser une batterie au plomb 6V de 4A couplé à deux panneaux solaire de 250 mA (valeur théorique loin de la réalité...) branchés en parallèle le tout géré par un régulateur de charge.

Il aurait surement été plus rentable niveau prix/capacité/taille d'utiliser des batteries au lithium. Mais ces dernières étant sensibles aux différences de température, à l'humidité et nécessitant des circuits de recharge plutôt compliqués, j'ai préféré rester sur une batterie classique au plomb.

Laisser les LEDs allumées en permanence entraîne une consommation bien trop importante (autours des 250mA) ce qui empêche d'alimenter le circuit sur batterie. La solution est donc de calibrer une fois le système sans courrier puis de faire des mesure à intervalles réguliers pour détecter la présence de courrier.

Ainsi le circuit consomme environ 75mA (non connecté à un réseau wifi), ce qui donne avec la batterie une autonomie d'environ deux jours. D'après mes mesures les panneaux solaires fournissent au maximum 300 mA en plein soleil mais cela chute au alentours de 100 mA avec un ciel nuageux. Ce qui sur une base de 10h de soleil par jour étend l'autonomie à environ 5 jours...

La solution pour ne pas avoir à recharger le système est donc d'utiliser une RTC et la fonction *sleep mode* de l'ESP. Cette fonction s'active en reliant la broche 4 (GPIO 16) à la broche 1 (reset) et au niveau du code avec:

ESP.deepSleep(temps en micro seconde);

Elle a pour effet de mettre en "sommeil" l'ESP8266 réduisant sa consommation au alentour de 1mA mais entraîne un reboot de ce dernier à son réveil. C'est ici qu'intervient la RTC, qui permet au système de se "repérer" dans le temps.

Connaissant à peut près les horaires de passage de mon facteur, j'ai décidé que l'ESP resterait allumé entre 10h et 14h et vérifierait sur cette plage horaire la présence de courrier toutes les 5 minutes. Le reste du temps (de 14h à 10h) l'ESP est en sommeil, et le circuit consomme environ 10 mA (consommation du à la RTC). Ce qui permet en théorie au système de fonctionner indéfiniment !!

En pratique le système à déjà fonctionné une semaine d'été sans problème. J'attend d'avoir encore un peut plus de recule pour être sur de son autonomie dans le temps











Étape 6 - Utiliser Twitter avec un ESP8266



Pour permettre à l'ESP8266 nous allons utiliser la Tweet Library:

https://arduino-tweet.appspot.com/

Ensuite pour pouvoir Tweeter vous allez devoir accorder l'accès au compte Twitter via une clef. Pour l'obtenir rendez vous sur cette page:

https://api.twitter.com/oauth/authorize?oauth_nonce=2828234892369209436&oauth_timestamp=1500815756&oauth_consumer_key=oQA2jr32rWowM4 SpGB64yQ&oauth_signature_method=HMAC-SHA1&oauth_version=1.0&oauth_token=DHDffgAAAAAAViGAAABXW-XDZc&oauth_signature=jPqM1kB NpqhVUIDJMFQn%2BB92uo4%3D

I Bien sur l'accès peut être supprimé a tout moment via vos paramètre twitter: https://twitter.com/settings/applications

Malheureusement la *Tweet Library* n'est pas nativement compatible avec les ESP car elle a été développée pour être utilisée avec un Arduino et un shield ethernet. Pour la rendre compatible il va vous falloir modifier le header (Twitter.h) Pour cela rajoutez ligne 15

#include <ESP8266Wifi.h>

passez en commentaire les lignes 21 et 23 et remplacez *EthernetClient* client; ligne 35 par

WifiClient client;





Cette application sera en mesure de :

- Lire les Tweets de votre fil.
- Regarder vos abonnements et suivre de nouvelles personnes.
- Mettre à jour votre profil.Publier des Tweets à votre place

Ne sera pas en mesure de :

- · Accéder à vos messages privés.
- Voir votre adresse email.
- Voir votre mot de passe Twitter.



Par NeoCat arduino-tweet.appspot.com/ Twitter Library for Arduino: post a tweet easily using Arduino

14	far nur nodfilih 🖸	In June 1	0
	0/1	1.5	
	Twitter.com - Anduino library to Fost messages to Twitter using Chath.		Twitte
	Copyright (c) HecCat 2010-2011, All right reserved.		Copyrie
	This library is distributed in the hope that it will be useful,		This 1
	but WITHOUT ANY WARRANTY, without even the implied warranty of		but WE
	MERCHARIABILITY or FITNESS FOR A FARTICILAR PORPOSE.	7	HERCEAL
	// ver1.2 - Tee (2dg.h) to support IDE 0019 or later	10	// ver1.
	// verl.3 - Duppert IDE 1.0	rarvnon	// ves1.
	Calcoler Delition a	- Jif (-	Sertions .
	PORTING INTERING	οαιπεε	
	tindule classing by		al and a state
	finitude (arr/commente b)		
	Odd defined department of hereiter > 11 // hereiter delt of hereit		ALC ONCO
	Alarinda (187.b)		Alachada
	tends f	20	Rendil I
	findlude (Tthemet.h)	23	//#inchu
	Haif defines(LADUINO) 11 LADUINO < 100 // earlier than Arduing 1.0	22 6	ALT OFTI
	#include (EthernetD85.h)	2.0	//#inclu
	- Jendi F	24	-Rendilf
		2.5	
	class Twitter	2.6	class Tw
	81	27 6	
	privater	2.6	priveter
	uint8_t parseStatus;	- 19	1580
	LNL statusCide/	30	144
	COLAT COAT +CIERK/	2.4	008.8
	Cart detree(TWDDIRD) If TWDDIRD < 100	2.2	art orto
	Clieft Clieft)		CLIM
	Tobarrat Clines, alient :		N
	August f		Annal A
	mahling	37	****
	Twitterdoonat that water and passed /	2.0	Twitt
		3.2	
	hool post(const char mang);	40	heel
	hool therkStatus(Frint +debug = MFLL) /	4.1	host
	int wait(Print *debug = NFLL)/	4.2	145
	Los status(void) (return statusCode;)	4.3	285
	-11		37
		4.5	
	Fendle //INITIER_H	4.6	#endif
		4.7	

3	
Twitter.opp - Arduing library to Fost messages to Twitter using GAuth. Copyright (c) NeoCat 2010-2011, All right reserved.	
This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCANATABLITY or FITHESS FOR A RAFICULAR FURPORE.	
// verl.2 - Use <ddg.ho 0019="" ide="" later<br="" or="" support="" to="">// verl.3 - Support IDE 1.0</ddg.ho>	Library
ALFRANK TRUTTER R	
Adding TWITTER #	modifióo
finglade (EIPEleWiFi.a)	mouniee
#include (introper.h)	
#include (pgsspace.h)	
#if defined(AADUINO) 44 AADUINO > 18 // Arduing 0019 or later	
finclade (SFLA)	
fendi.f	
//#Include (Ethernet.m)	
FLE OFFICE(ANDVING) && ANDVING V 100 77 EATLEFT THAN ADDIEDD 1.0	
Internation Content Content	
class Twitter	
privates	
uinté_t perseStatus/	
ist statusCode/	
CORAL CRAF *LORES/	
Clima alignet and the analysis of the	
false	
BifiClient client/	
fendif	
publics	
Twitter(const char *sser_and_passed) /	
bool postfronst char Amegh/	
hool checkStatus(Frint +dekug = MULL) /	
int wait(Print *debug = \$5513)	
<pre>Ltt status(rold) (return statusCode;)</pre>	
17	
feeded (Charters a	
Provide An and a second s	

Étape 7 - Programmation de l'ESP8266

Pour programmer l'ESP82266 on va uiliser l'IDE Arduino. Pour cela, il vous faut tout d'abord ajouter le support de ce type de carte dans l'IDE en ajoutant l'URL suivante dans Fichier>Préférences>URL de gestionnaire de cartes supplémentaires :

http://arduino.esp8266.com/package_esp8266com_index.json,http://arduino.esp8266.com/stable/package_esp8266com_index.json

Puis choisir "Generic ESP8266 Module " dans Outils>types de carte. Et enfin relier l'ESP au convertisseur USB/série comme indiqué sur le schéma. Téléchargez et installez ensuite la librairie RTC développée par Makuna pour gérer la DS3231:

https://github.com/Makuna/Rtc

Puis exécuter le programme DS3231_Simple pour mette à l'heure la RTC.

Et enfin vous pouvez envoyer sur l'ESP8266 le programme suivant pour gérer votre boite aux lettre connectée (Cf algorigramme pour les détails sur son fonctionnement):

https://drive.google.com/open?id=0B8tCTkPLfNNraHhlbnNaTmx3VIU

Pensez à adapter les plages de fonctionnement en fonction de vos besoins





Echer Editor Croquis Ouglis Ade					Þ	
notifiuto						
Serial.println("Wifi connected"):					6	
// Start the server						
server.begin(D)						
Serial println("Server started"); // Print the IP address Serial, println(biffi.localDP(D);	Préférences Paramètres Báseau					
simple basis(): // initiality the MacRicel library	Emplacement du camet de croox	15				
ierial.printle.Code Setup's:	(met/Archine			Parrowstr		
3	Choix de la lanque :	Lanque du sustière	Inicessite un red/marra	age of Andultan)		
veod loop D (Talle de police de l'éditeur	12				
if (mal 1) (//Télegram purple	Interface scale:	Automatique 100 : % (nécess	ite un redémarrage d'Arduino)			
de lay (2000);	Afficher les résultats détaillés	🔿 1181 da cantinaceira da castas maniferra	atalaat			
if (ma) 20 (//2001 red	Avertissement du compliateur					
notif(200.10.0); onlay(2000);	O Afficher les numéros de lig	rores les ous sopplemental es, one par ogne etp://archino.espit2i4i.com/stable/package.esi	p8266com_index3son			
	C Activer le repli de code					
<pre>if { wal == 30 { //imap yellox notif(200, 100, 0);</pre>	Vérifier le code après télév					
de 1 ay (2000) :	Winfler les mises à jour aux	liquez pour la liste non-officielle des URL des ca	rtes supportées			
if (wal 40 (//Twitter blue	😸 Mettre à jour l'extension d		OK Annuler			
tetif(0, 0, 200); (elay(2000);	Sauvegarder pendant la vé					
A Contraction of Contraction	URL de gestionnaire de cartes sup	plémentaires [http://arduino.esp8266.com/st	able/package_espli266com_index.json			
retif(50, 200, 0);	Dawrtage de préférences pouvert être édities directement dans le fichier					
)		ho ne s'exécute pas)				
<pre>if { xal -= 60 { //Wessenger light blou retif(0. 100. 100);</pre>						
A first and the first fall and a first				OK Annuler		
rainbow(20):						
3						
if [wal -= 60 motif(0. 0. 00; //turn off all lods					6	
Erregistrement terminé.						
					i i	
					1	
145			54941	STREES MARKE BOMME BOMME COL. 115200, 5124 (SHE SPORTS)	. d. Disabled, York av /deutty/1583	

Étape 8 - Conclusion

Ce tutoriel arrive à son terme, j'espère avoir été clair dans mes explications. Cependant, si jamais vous ne comprenez pas un point n'hésitez pas à engager la conversation ③ En tous cas si vous reproduisez ce système ou que vous avez des idées d'amélioration partagez les !!!

J'ai choisi d'utiliser Twitter comme moyen de notification mais il est tout a fait envisageable de charger ce point. Je pense par exemple à une lumière allumé sans fil, ou un système de requête http ou même un couplage avec un projet domotique comme https://gladysproject.com/fr/ ©





Notes et références

Hackable magazine numéro 2 et 7 ESP8266:

- http://arduino.esp8266.com/versions/1.6.5-1160-gef26c5f/doc/reference.html
- http://esp8266.github.io/Arduino/versions/2.0.0/doc/boards.html.
- http://esp8266.github.io/Arduino/versions/2.0.0/doc/libraries.html
- http://www.esp8266.com/wiki/doku.php?id=esp8266_power_usage
- https://www.losant.com/blog/making-the-esp8266-low-powered-with-deep-sleep

Tweet Library:

- https://arduino-tweet.appspot.com/
- https://gist.github.com/hofmannsven/6226895
- https://hofmannsven.com/2013/laboratory/arduino-twitter-library/

Choix de la batterie:

https://www.youtube.com/watch?v=LqgP16JQ24I