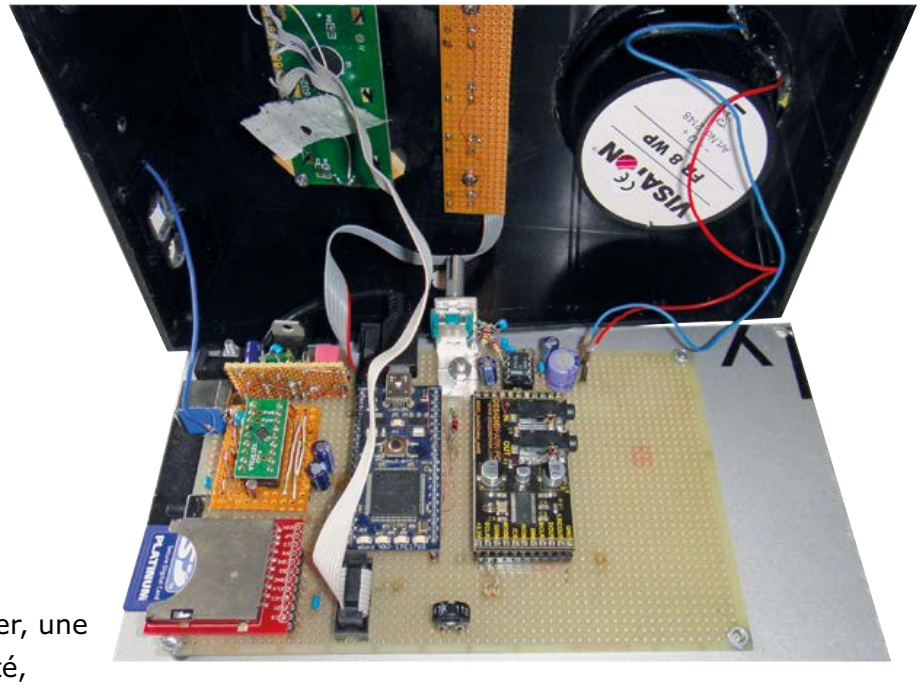
écouter la radio en différé

FM, j'te play plus, j'te rembobine[1]



Hans Oppermann

Vous avez tous connu ça : au petit-déjeuner, une information à la radio suscite votre curiosité, mais vous n'êtes pas assez bien réveillé pour la retenir. Ah ! Si vous pouviez rembobiner… Justement, ce projet enregistre le flux radiophonique dans une mémoire circulaire de manière à pouvoir le réécouter à la demande. Il repose sur une radio, un codec audio et une petite carte à processeur ARM.

Sur la télévision, satellitaire ou câblée, vous disposez de fonctions comme pause, retour en arrière, lecture en différé, saut vers le direct, etc. Mais pour la radio, je n'ai rien trouvé de semblable. Mon principe est simple dans ce cas-là : tu n'en as pas, alors construis-le !

De l'idée…

Le module mbed [1] avec processeur ARM LPC1768 que j'avais au laboratoire ne me servait que pour des expériences. J'appréciais sa puissance de calcul et la profusion des entrées/sorties. Mais là, je lui avais enfin trouvé un projet à sa mesure, j'en ai fait un enregistreur du signal de la radio qui permet la restitution simultanée d'un passage antérieur. Je vous explique comment.

En fouinant sur la page de mbed-Cookbook [2], je suis tombé, sous la rubrique Audio, sur le parfait article intitulé *TLV320AIC23B – un codec audio à interface I2S peut jouer et enregistrer !* [3]. « Jouer et enregistrer », les deux à la fois : quelle trouvaille ! Plus j'y pensais, plus je me demandais s'il était vraiment possible, avec ce codec, de faire les deux en même temps. Ioannis Kedros et Daniel Worral, les concepteurs de la bibliothèque logicielle, m'ont confirmé que c'était le cas. Qu'ils en soient remerciés !

Un codec, contraction de codeur-décodeur, transforme des flux ou des signaux en vue de leur transmission ou de leur enregistrement. Je n'avais jamais entendu parler de ce codec TLV320AIC23B alors que cette puce de Texas est sur le marché depuis plus de dix ans ! La puce que j'utilise est montée sur une carte d'expérimentation, qui n'est pas l'originale de Texas Instruments, mais un petit circuit imprimé de DesignSpark. Les lecteurs d'Elektor connaissent bien le logiciel gratuit de dessin de circuits imprimés DesignSpark PCB. Ce logiciel est développé par le distributeur de composants *RS Components* qui propose la carte en question, mais malheureusement uniquement via RS [4]. Le reste des composants ne m'a pas donné de soucis particuliers.

à la construction modulaire

Mais que fait donc vraiment la pièce maîtresse du circuit, le module AudioCodec ? Il convertit les signaux audio stéréo en signaux numériques sériels et aussi l'inverse. Pour ce faire, il utilise pour le transfert des données le protocole I2S (à ne pas confondre avec I²C) développé par Philips et que l'**encadré** vous fait découvrir. Avec lui, le module AudioCodec est capable d'enregistrer et de reproduire des contenus audio dans diverses résolutions et avec différents taux d'échantillonnage. Comme récepteur (voir **figure 1**), on retrouve une vieille connaissance, le Si4735 de Silicon Labs. Ce récepteur de radio complet empaqueté dans un boîtier SSOP n'est plus fabriqué par Silicon Labs, mais on le trouve toujours chez de nombreux distributeurs sur une carte d'adaptation DIL. Le Si4735 délivre (par les condensateurs de couplage C1 et C6) les signaux stéréo analogiques de sortie ROUT et LOUT, comme le veut le module de codec audio.

La puce de codec audio en tire le signal numérisé DOUT qui est envoyé au module fonctionnel I2S du processeur. Lequel stocke le signal sur une carte SD, qui constitue une mémoire circulaire d'un volume de 1 Go, et le renvoie par la ligne DIN au codec audio. Enfin, le signal analogique aboutit sur l'amplificateur audio formé essentiellement d'un LM386 pour la restitution du son dans le haut-parleur. On réalise la fonction de différé en enregistrant le flux en temps réel et en le

[1] *d'après Philippe Léotard*

lisant sélectivement. Quand on prend le direct, les données sont identiques ; en différé, elles correspondent à un autre moment du passé. Nous y reviendrons dans la description du logiciel.
Les autres composantes du circuit servent à la commande d'accord du récepteur et de l'accès aux stations, à celle du différé et à l'affichage des données utiles sur le LCD.
Une mention spéciale pour le RPG, le générateur d'impulsions rotatif, qui n'est exploité que partiellement, pour le réglage du volume et la commande par menu seulement. Pour les autres fonctions, en particulier le différé, il faut encore quelques boutons-poussoirs.
Le module de mise sous et hors tension est assez peu conventionnel, mais indispensable. Le processeur doit être prévenu quand on veut éteindre la radio avec S5. Pas question de simplement retirer la fiche de la prise, il faut au préalable enregistrer sur la carte SD des données telles que le réglage des boutons de station, la dernière fréquence et le dernier niveau de volume réglés. Le processeur est averti du débranchement par P27, il stocke alors les données et finalement coupe lui-même le courant, en configurant P27 en sortie, pour ensuite le mettre à zéro. Lors de l'enclenchement par S5, l'arrivée de la tension d'alimentation fait démarrer le processeur. Il relit les données enregistrées de manière à ce que tout redevienne comme avant, station sélectionnée et volume réglé. Ce circuit, je ne l'ai pas inventé tout seul, il est basé sur [6].

Voyons maintenant comment régler le récepteur à l'aide des fonctions du menu qui s'affiche sur le LCD quand on pousse sur S4.

Select Station
Dans ce sous-menu, on sélectionne un des émetteurs affectés aux chiffres 1 à 6.

Dial Frequency
Le RPG permet de régler la fréquence de réception de 87,50 MHz jusqu'à 108 MHz, et retour.

Set Station
La fréquence actuelle d'accord se voit attribuer un numéro de station entre 1 et 6.

Exit Menu
Pour quitter le menu.

Pour activer l'option choisie dans le menu, j'avais prévu d'utiliser le RPG, en appuyant sur le bouton, mais ce n'était pas pratique. Mieux vaut attribuer cette fonction supplémentaire au bouton-poussoir S5 (*On/Off*). Quand le menu est actif, il sélectionne l'option, sinon, en service normal, il allume ou éteint l'appareil.
Pendant les opérations sur le menu, il faut interrompre la restitution de la radio pour ne pas surcharger le processeur. Le bouton-poussoir S3 (*Search*) permet, en cours d'écoute, de rechercher une autre station. On en règle le volume avec le RPG.
Et l'écoute en différé ? On la manœuvre avec les boutons S1 (*Start/Stop*) et S2 (*Live*) ainsi que le RPG. Appuyer sur S2 arrête la restitution. Les secondes enregistrées jusque-là sur la carte SD sont affichées sur le LCD. Le RPG permet de « rembobiner ». Une action sur S1 ramène la lecture à l'endroit du retour en arrière pour une nouvelle écoute. S2 met fin à l'écoute en différé et ramène au direct.

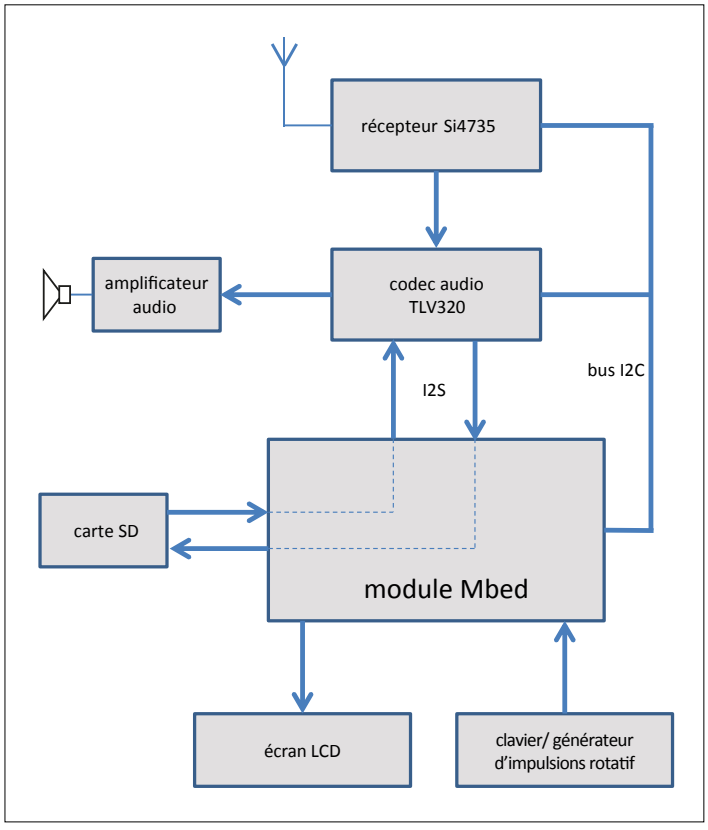


Figure 1. Diagramme fonctionnel de la radio à écoute en différé, celle qui rembobine

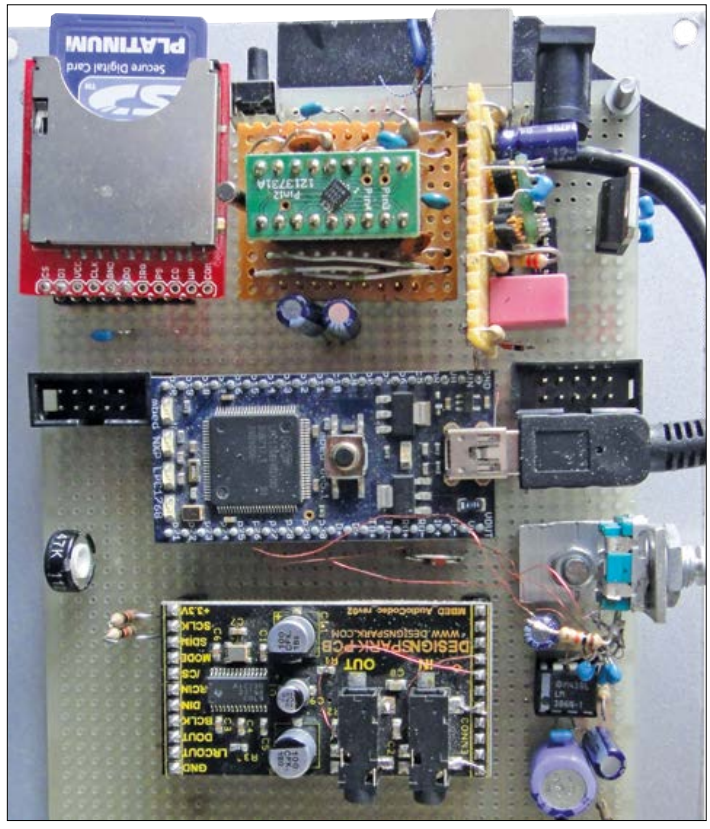
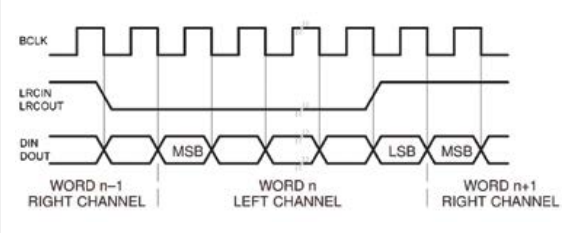


Figure 2. Platines sur platine – ici tout est modulaire !

Interface I²S



source: Philips

Les données audio numériques sont transmises à travers des interfaces synchrones de puce à puce (*Inter IC Sound*), le plus souvent entre convertisseurs A/N ou N/A et processeurs de signaux numériques. Ces interfaces disposent de deux lignes de signal (DIN et DOUT), deux lignes de commande pour l'enregistrement et la restitution (LRCIN et LRCOUT) ainsi que d'une ligne d'horloge BCLK. Les lignes Word-Select LRCIN et LRCOUT déterminent la chronométrie des signaux de données et la répartition des signaux sur le canal audio. La fréquence des signaux Word-Select correspond à la vitesse d'échantillonnage des signaux. Traditionnellement, on utilise la modulation par impulsion et codage (MIC), ou PCM en anglais (*Pulse Code Modulation*), mais d'autres formes de codage existent. Le signal PCM transmet toujours deux canaux audio en alternant chaque mot de données du canal gauche et du canal droit.

Logiciel du différé

La partie du logiciel que j'ai rédigé est incluse dans un seul module, *main.cpp*, disponible sur [7]. Toutes les autres parties sont des fonctions de bibliothèques du projet mbed. Je me limite donc ici à expliquer ce module.

En tête du module, il y a les bibliothèques pour reconnaître les composantes matérielles employées. À chaque composante, on attribue une instance d'objet. Suivent les déclarations des différentes variables et un grand nombre de fonctions d'aide, le tout bien documenté dans le code source. Dans la fonction principale *main* elle-même, on fait l'initialisation successivement du LCD, du menu de fonctions, de la carte SD de 1 Go et la lecture des données de configuration décrite ci-dessus.

On affecte aux boutons avec la fonction `initButtons()` une fonction d'interruption appelée lors de chaque action sur le bouton. Vient ensuite l'initialisation du récepteur, du TLV320 et du RPG. On met ainsi le TLV320 en mode BOTH, ce qui fait qu'il peut enregistrer et lire en même temps. En outre, on ajoute une interruption audio par la fonction `record_play` qui est appelée quand le tampon audio pour les données reçues est plein.

La boucle *while* suivante (cf. **listage**) assure la fonction de différé au moyen des deux variables entières `read_sector` et `write_sector`. C'est avec `write_sector` que l'on écrit les données sur la carte SD et avec `read_sector` qu'on les lit pour les envoyer à l'amplificateur audio.

Listage : fonction d'écoute en différé

```
while(recording) {
   streamToSD();
   if ((read_sector < write_sector) && playing) fillBuffer();
                                  //continually fill circular buffer
   if (playing) {                 // no pause
       if (iValue > lastRPG) {Vol--; audio.outputVolume(&Vol);}
       if (iValue < lastRPG) {Vol++; audio.outputVolume(&Vol);}
   } else {
       if (iValue > lastRPG) rewindPlaying();
       if (iValue < lastRPG) forwardPlaying();
   }
   lastRPG = iValue;
}
```

Pousser sur le bouton *Stop* S1 met dans la fonction `pausePlaying()` la variable `playing` sur `false`. Ce qui arrête la reproduction, puisque le tampon `fillBuffer()` ne sera plus appelé. On peut alors changer dans `rewindPlaying()` et `forwardPlaying()` le `read_sector`. Si vous appuyez de nouveau sur S1, `playing=true`, la lecture peut reprendre, en différé, dans le `read_sector` modifié ou être répétée. En poussant sur le bouton *Live* (S2) on impose `read_sector=write_sector` pour retourner au direct.

Finalement, on peut aussi décrire le logiciel comme suit. Avec `record_play()`, les données audio sont enregistrées et écrites dans le `write_sector` correspondant sur la carte SD. La fonction `fillBuffer()` récupère les données audio dans le `read_sector` de la carte SD et les sort vers l'amplificateur audio. Dans l'intervalle, s'il le souhaite, l'utilisateur peut modifier le `read_sector` avec la fonction d'écoute en différé.

Le logiciel a été développé sur la page mbed, il est aussi compilé en ligne et peut être chargé par USB comme objet dans le module mbed. On peut lire sur cette page les détails sur l'ensemble du processus de développement avec un module mbed.

La construction

J'ai construit la radio à écoute en différé sur une plaque à trous qui rassemble les différents modules et leur câblage. Dessiner un circuit imprimé ne servirait à rien. La **figure 2** montre comment je les ai organisés. Au milieu, le module mbed, en dessous la communication avec le LCD, au-dessus, le connecteur pour le clavier et, à droite, le module codec audio. À gauche, en bas, le support pour la carte SD et au-dessus le récepteur Si4735 qui se trouve sur un petit circuit imprimé DIL mais a aussi besoin de quelques composants externes. Voilà donc le récepteur complet, modulaire à souhait, construit sur une plaque à trous — la section On/Off en haut, à gauche, a été montée à la verticale.
Sur le bord gauche de la plaque perforée, il y a aussi les prises pour l'alimentation, l'antenne, le bouton de mise à zéro et le port USB. Un câble USB raccourci relie cette prise au module mbed. À part quelques composants isolés, il n'y a plus que l'amplificateur audio au-dessus à gauche, formé de quelques composants

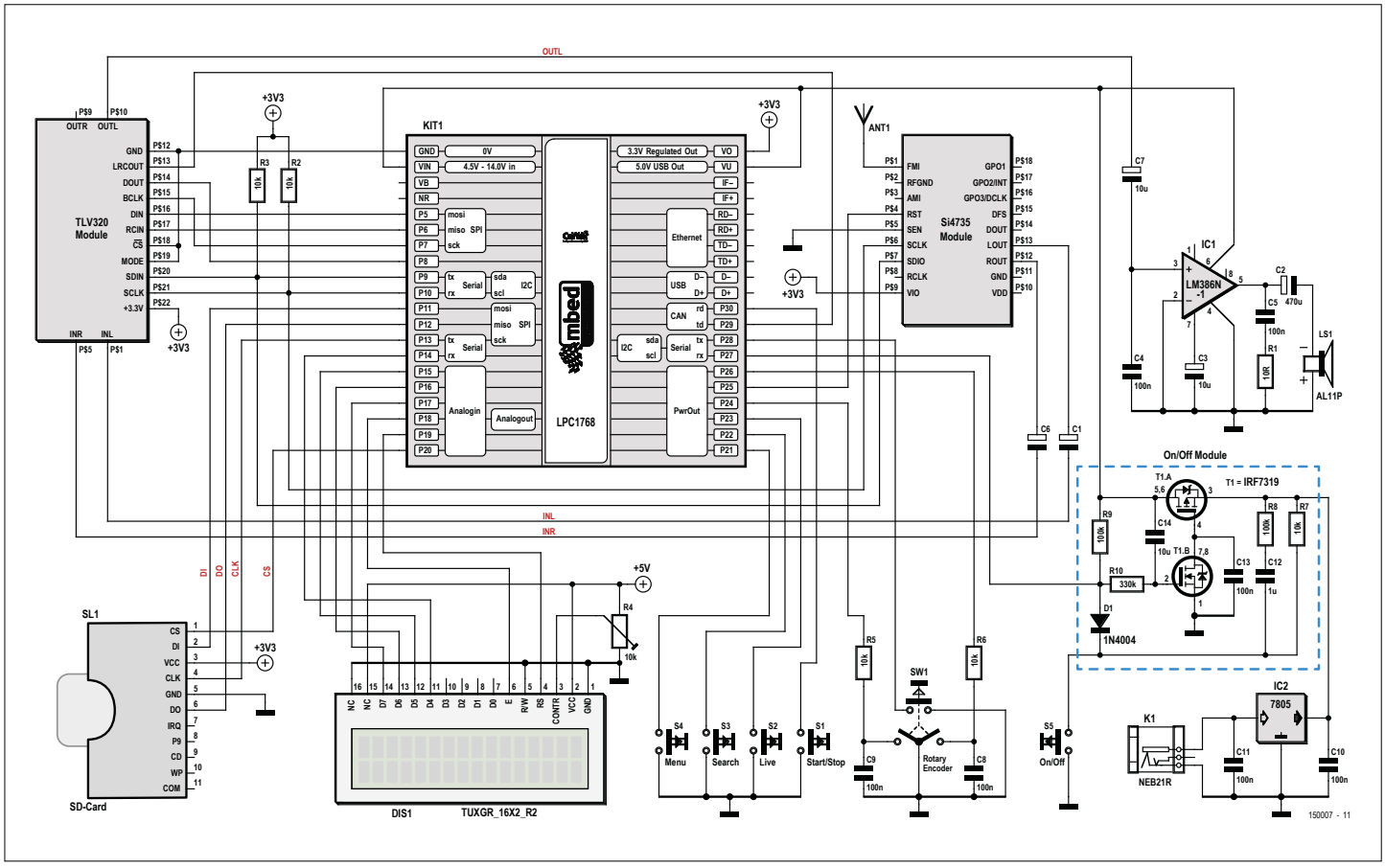


Figure 3. Comment relier tous ces modules entre eux, c'est le schéma qui vous le dira.

discrets. Oui, discret, comparé aux autres sections, on peut aussi considérer le LM386 comme tel.

Que reste-t-il à voir sur la plaque perforée ? Le codeur rotatif avec les condensateurs C8 et C9 soudés dessus, les résistances R5 et R6 à proximité de l'ampli audio, un potentiomètre d'ajustage en dessous pour régler le contraste du LCD, les résistances de rappel haut pour le bus I²C en bas du module mbed, au-dessus du module On/Off, le régulateur de tension 7805 avec ses condensateurs électrolytiques de découplage et enfin, à droite près du récepteur, les condensateurs de couplage pour le module AudioCodec. Rien ne manque à l'appel ? Munissez-vous de la photo du circuit (**figure 3**) pour localiser chaque composant.

Le câblage m'a paru assez laborieux, il a consisté à bricoler les liaisons avec un outil manuel à wrapper (en câblage enroulé) et du fil de cuivre de 0,2 mm. L'embarrassante question de l'habillage, je l'ai éludée en utilisant un boîtier noir en forme de pupitre, comme on le voit sur l'illustration en début d'article. Pas particulièrement élégant, mais ça marche ! Indépendamment de l'esthétique extérieure, j'imagine une amélioration possible dans le domaine de la sensibilité du récepteur. Même si de nombreuses descriptions du module Si4735 assurent qu'il a une bonne sensibilité, je ne partage pas ce point de vue. C'est probablement dû à mon agencement ou au boîtier. Mais le circuit n'est nullement dépendant du Si4735 comme récepteur. Il est tout à fait possible d'aménager une entrée audio pour un poste existant ou mieux, un récepteur mondial. Pour l'anti-rebond des boutons-poussoirs aussi, il y a mieux à faire. Mais en définitive, il s'agit encore ici d'une ébauche brute, libre à vous d'y apporter votre savoir-faire et même votre fantaisie. ⏮

(150007 – version française : Robert Grignard)

Liens

[1] https://developer.mbed.org/

[2] https://developer.mbed.org/cookbook/Homepage

[3] https://developer.mbed.org/cookbook/TLV320AIC23B

[4] Circuit DesignSpark, réf. RS 754-1974, disponible sur le site RS des professionnels et celui des particuliers
Fiche technique du circuit DesignSpark : http://docs-europe.electrocomponents.com/webdocs/10fc/0900766b810fcde6.pdf

[5] www.ak-modul-bus.de/cat/documentation/Si4735.Documentation.zip

[6] www.mosaic-industries.com/embedded-systems/microcontroller-projects/electronic-circuits/push-button-switch-turn-on/microcontroller-latching-on-off

[7] www.elektormagazine.fr/150007