

Systemy Mikroprocesorowe 2

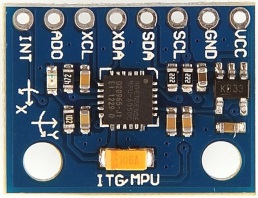
Łukasz Motyka, Filip Hepko

Smart Surface

Celem naszego projektu było napisanie programu który wykorzystując magistralę I2C zaimplementowaną w procesor ARM Cortex M0+ na płytce ewaluacyjnej Kinetis L 25z, i odczytywanie danych z zewnętrznego akcelerometru na układzie MPU6050. Wykorzystaliśmy wbudowaną jednostkę Motion Proccessing Unit do wychwytywania „puknięć” jeśli układ leży na jakiejś powierzchni (drgania są przenoszone przez daną powierzchnie i rejestrowane przez akcelerometr). Inteligentność urządzenia polega na odróżnianiu dwóch szybkich puknięć w powierzchnie od dwóch wolnych. W naszym projekcie na dwa szybkie puknięcia zmienia się stan diody czerwone a na dwa wolne diody zielonej. Do tego inicjujemy jednostkę MPU, aby generowała przerwania gdy wystąpią drgania. Wartości do inicjalizacji dobrane zostały eksperymentalnie. Następnie generowane przerwanie przez MPU jest zaimplementowane jako przerwanie od GPIO na płytce KL25Z. W obsłudze przerwania włączamy timer PIT do zliczania czasu między puknięciami. Jeśli natomiast wystąpi tylko 1 puknięcie PIT się przepełnia po 3 sekundach i cały algorytm się resetuje.

W naszym projekcie wykorzystaliśmy:

 Płytkę ewaluacyjną FRDM-KL25Z

 Akcelerometr na MPU-6050

 Środowisko programistyczne KEIL v5

Nasz program oprócz main.c, plików obiektowych i reszty plikow generowanych przez Keila ma:

* MPU6050.c – konfiguracja MPU, zawiera funkcje:
  + MpuInit – ustawienie odpowiednich bitów w rejestrach potrzebnych do startu układu
  + SetRangeAccel – ustawiamy zakres akcelerometru
  + ReadAxisAccel – odczytujemy surowe dane z każdej osi akcelerometru
  + ReadTemperature – odczytujemy surowy pomiar temperatury i przeliczamy na zrozumiały sposób
  + MotionInit – inicjalizujemy wewnętrzną jednostkę obliczeniową
* i2c.c – inicjalizacja i2c
  + ReadRegisterI2C
  + WriteRegisterI2C
* pit.c – jako główny timer w programie decydujący o rozpoznawaniu puknięć, i w tym pliku go konfigurujemy
  + pitInit
  + PIT\_IRQHandler
* uart.c – gdy testowaliśmy to przesyłaliśmy dane przez port szeregowy aby zobaczyć surowe dane z akcelerometru
  + uart\_init
  + SendString
  + GetValue