FPV Headtracker inercyjny sterownik ruchu (2)

Postęp technologiczny sprawił, iż wynalazki, które kilkadziesiąt lat temu budziły podziw i zachwyt, dziś są w normalnym użytkowaniu. Na przykład, w napięciu oglądało się filmy, w których pokazywane były zaawansowane „zachodnie technologie". Przykładem może być system Helmet-Mounted Display stosowany w lotnictwie wojskowym już w latach siedemdziesiątych. Systemy HMD szybko stały się standardem i nadal umożliwiają pilotowi podglądanie informacji na ekranach wbudowanych w hełm lotniczy, a oprócz tego dają też możliwość sterowania wybranymi funkcjami maszyny. Znamy systemy sterowania karabinem podczepionym do śmigłowca Apache za pomocą ruchów głowy pilota, gdzie cel jest namierzany bez konieczności odrywania uwagi od pilotażu maszyny. Kolejnym etapem rozwoju tej technologii jest świat „Artificial Reality" (AR), który możemy obserwować poprzez specjalne okulary, jak np. Oculus Rift, lub na ekranach telefonów komórkowych. Użytkownicy tych urządzeń wiedzą, iż efekt jest uzyskiwany dzięki wykorzystaniu pozycjonowania i żyroskopu, który mierzy kąty pochylenia w danej pozycji. Niewielu z nich jednak wie jak to w istocie działa. Dzięki miniaturyzacji i stosunkowo niskiej cenie technologii MEMS możemy zbudować podobny system we własnym warsztacie.

Rekomendacje: *urządzenie przyda się nie tylko do modelu, ale również do innych zastosowań, takich jak: monitoring obiektu, robotyka, urządzenia dla osób niepełnosprawnych i innych.*

Filtr Kalmana

Wyobraźmy sobie, że stoimy na ulicy i trzymamy w ręku GPS. Mimo że nasza pozycja jest niezmienna, to odczyty w każdej sekundzie są różne, ale mieszczą się w pewnym obszarze (np. w okręgu o promieniu 5 metrów). Chcielibyśmy „uśrednić" odczyty z GPS w taki sposób, aby uzyskać właściwą pozycję. Aby tego dokonać musimy na podstawie wskazań z GPS przewidzieć faktyczne miejsce postoju. Dokładnie na tym polega filtrowanie Kalmana – na estymowaniu najbardziej prawdopodobnego (z minimalną wariancją) stanu układu (wyniku pomiaru). Szukamy estymacji pozycji $\hat{x}_k$ w chwili **k** na podstawie pomiarów z GPS (Z_k), poprzedniej estymacji $\hat{x}_{k-1}$ oraz współczynnika wzmocnienia Kalmana K_k.

Rozpoczynamy od zdefiniowania modelu matematycznego naszego układu za pomocą następujących równań:

$$x_k = A \cdot x_{k-1} + B \cdot u_k + w_{k-1}$$
$$z_k = H \cdot x_k + v_k$$
$$w_k \sim N(0, Q_k)$$
$$v_k \sim N(0, R_k)$$

gdzie:

- x_k oznacza stan układu w chwili k, który jest zależny od poprzednio zarejestrowanego stanu, sygnału sterującego u_k oraz szumu przetwarzania w_k.
- A jest macierzą przejść (systemu),
- B jest macierzą sterowania (wejścia).

Drugie równanie przedstawia obserwowany pomiar z_k jako zależność pomiędzy faktycznym stanem systemu x_k i wektorem szumu pomiarowego v_k. Teoretycznie, faktyczny stan systemu x_k nie jest znany w żadnym momencie. Możemy go jedynie obserwować poprzez pomiary z_k, które są niedokładne.

H jest macierzą wyjścia, wykorzystywaną w wypadku zamiany (mapowania) wyliczanych wartości na ich odpowiedniki w układzie obserwowanym (realnym). Przyjmuje się, że szumy w_k i v_k są o rozkładzie normalnym, o kowariancjach Q_k i R_k. Zakładamy także liniową zależność pojawiających się po sobie pomiarów. Algorytm iteracyjnie wylicza estymatę stanu w chwilina podstawie wartości w chwili *k-1*, co jest charakterystyczne dla modelu dyskretnego.

Filtrowanie Kalmana przebiega dwuetapowo. Pierwszym etapem jest faza predykcji (aktualizacji czasu):

$$\hat{x}_k^- = A \cdot \hat{x}_{k-1} + B \cdot u_k$$
$$P_k^- = A \cdot P_{k-1} \cdot A^T + Q_k$$

W pierwszym równaniu przewidywana jest estymata stanu $\hat{x}_k$. W drugim jest obliczana kowariancja zmian estymaty P_k, czyli „dokładność" estymaty Q_k jest macierzą estymacji kowariancji błędów P_k. Możemy to także rozumieć, jako zaufanie do przewidzianej estymaty. Kolejny etap – faza korekcji – wygląda następująco:

$$K_k = \frac{P_k^- \cdot H_k^T}{H_k \cdot P_k^- \cdot H_k^T + R_k}$$
$$\hat{x}_k = \hat{x}_k^- + K_k \cdot (X_k^- - H_k \hat{x}_k^-)$$
$$P_k = (1 - H \cdot K_k) \cdot P_k^-$$

W tej fazie obliczane jest wzmocnienie Kalmana K_k. Następnie, na jego podstawie dokonuje się aktualizacji estymaty stanu i kowariancji predykcji, które zostaną użyte w kolejnej iteracji algorytmu. R_k jest to estymacja kowariancji szumów wzmocnienia K. Na pierwszy rzut oka teoria wydaje się bardzo zawiła, jednak staje się zrozumiała, gdy wykorzystamy ją w praktycznych doświadczeniach.

W najprostszym wypadku, jeśli chcemy filtrować pojedynczą skalarną wartość pomiaru (np. czas kanału PPM) wzory ulegają znacznemu uproszczeniu. Macierz przejść A staje się liczbą 1, pomijamy sterowanie układem, a zatem u_k=*0*. Faza predykcji przybiera postać:

$$\hat{x}_k^- = \hat{x}_{k-1}$$
$$P_k^- = P_{k-1} + Q$$

Macierz wyjścia H również przyjmuje wartość 1, ponieważ interpretacja pomiarów (czasy kanału PPM) jest jednakowa w modelu matematycznym, jak i w obserwacjach.

$$K_k = \frac{P_k^-}{P_k^- + R}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{x}_k^-)$$
$$P_k = (1 - K_k)P_k^-$$

Pozostały nam tylko 2 zmienne skalarne Q i R, za których pomocą możemy w łatwy sposób sterować „siłą" filtrowania. Poprzez odpowiednie dobranie parametrów nasz filtr będzie w mniejszym lub większym stopniu reagował na szumy pomiarowe. Implementacja powyższego filtru zawiera się w kilku liniach kodu:

```
double filtrKalmana(double zk,
double *xk, double *Pk) {
  //double xk_1 = *xk;
  //double Pk_1 = *Pk;
  //Faza predykcji
  //*Xk = Xk_1;
  *Pk = *Pk + (double)Q;
  //Faza korekcji
  double Kk = *Pk / (*Pk +
(double)R);
  *xk = *xk + Kk*(zk - *xk);
  *Pk = (1 - Kk)*(*Pk);
  //Zwrot nowej estymaty
  return *xk;
}
```

Sposób wykorzystania w programie jest następujący:

```
estymata = filtrKalmana(pomiar,
&xk, &Pk);
```

Funkcja zwraca nową estymatę stanu. W parametrach przekazujemy aktualny pomiar, poprzednią estymatę x_k i kowariancję *Pk*. Jako że x_k i *Pk* biorą udział w następujących po sobie iteracjach, ich wartości muszą być zapamiętywane w każdym wywołaniu funkcji, stąd przekazanie przez wskaźnik. Liczby Q i R (kowariancje szumów) zostały zdefiniowano poza ciałem funkcji. Tym sposobem za pomocą niewielkiego kodu i 2 zmiennych dostajemy do dyspozycji jedno z najlepszych narzędzi do „wygładzania" wszelkich przebiegów linowych. Z powodzeniem możemy nim zastąpić powszechnie używaną metodę uśredniania wyników.

Wracając do headtrackera – poprzez dodatkowe filtrowanie czasów kanałów PPM, uzyskujemy efekt płynnych ruchów kamery, co ma wpływ na dłuższą żywotność serwomechanizmów oraz może się także okazać przydatne przy filmowaniu z powietrza. Na **rysunku 14** pokazano przykładową serię szybko zmieniających się pomiarów z akcelerometru. Czerwona linia to efekt działania filtru Kalmana, natomiast kolorem czarnym pokazano średnią ruchomą rzędu 20. Jak widać czasy reakcji na zmiany pomiarów, a także estymaty stanów zostały poprawione.

Podstawowe informacje:

- Demonstracja działania: *http://goo.gl/Wqhefq*
- Bazuje na akcelerometrze LM9S oraz na mikrokontrolerze 8-bitowym (ATmega8 lub ATmega168).
- Współpracuje z dowolną aparaturą do zdalnego sterowania.
- Wykorzystanie dwóch kanałów do sterowania ruchem dwóch serwomechanizmów.
- Wariant podstawowy jest przeznaczony do modeli samolotów.

Dodatkowe materiały na FTP:

ftp://ep.com.pl, user: 87550, pass: rxoaagj8

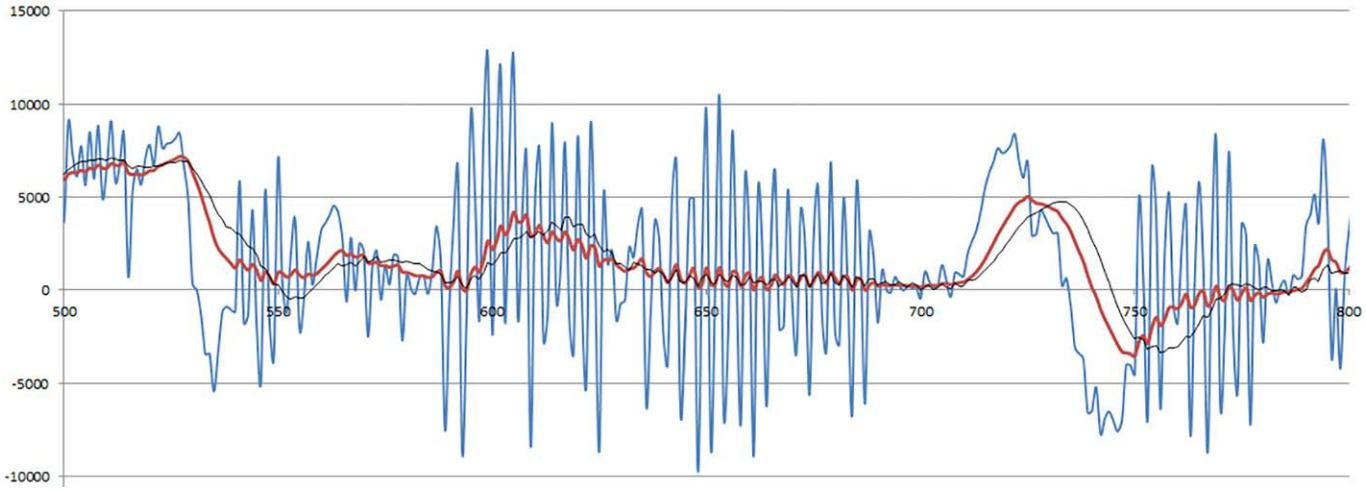
- wzory płytek PCB

Projekty pokrewne na FTP:
(wymienione artykuły są w całości dostępne na FTP)

AVT-5491	Czujnik inercyjny LSM9DS0 oraz jego zastosowanie praktyczne (EP 2/2015)
AVT-1806	Detektor drgań (EP 7/2014)
AVT-5393	ASO – Automatyczny system ostrzegania (EP 4/2013)
AVT-5387	gLogger – 3-osiowy rejestrator przyśpieszenia (EP 3/2013)
AVT-5223	Kieszonkowy akcelerometr (EP 2/2010)
Projekt 132	Miernik przyśpieszenia (EP 8/2005)

* Uwaga:
Zestawy AVT mogą występować w następujących wersjach:
AVT xxxx UK to zaprogramowany układ. Tylko i wyłącznie. Bez elementów dodatkowych.
AVT xxxx A płytka drukowana PCB (lub płytki drukowane, jeśli w opisie wyraźnie zaznaczono), bez elementów dodatkowych.
AVT xxxx A+ płytka drukowana i zaprogramowany układ (czyli połączenie wersji A i wersji UK) bez elementów dodatkowych.
AVT xxxx B płytka drukowana (lub płytki) oraz komplet elementów wymienionych w załączniku pdf
AVT xxxx C to nic innego jak zmontowany zestaw B, czyli elementy wlutowane w PCB. Należy mieć na uwadze, że o ile nie zaznaczono wyraźnie w opisie, zestaw ten nie ma obudowy ani elementów dodatkowych, które nie zostały wymienione w załączniku pdf
AVT xxxx CD oprogramowanie (nieczęsto spotykana wersja, lecz jeśli występuje, to niezbędne oprogramowanie można ściągnąć, klikając w link umieszczony w opisie kitu)
Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! (UK, A, A+, B lub C). *http://sklep.avt.pl*

Teoria Kalmana ma znacznie większe zastosowania, począwszy od powyższych, poprzez systemy telemetryczne w Nascar czy Formule 1, po systemy sterowania w przestrzeni kosmicznej. W Internecie dostępnych jest wiele publikacji opisujących różne zastosowania filtru. Wśród nich jest także rozwiązanie, które możemy wykorzystać zamiast filtru komplementarnego. Opisane zostało na stronach [1] oraz [2] w wymienionych poniżej w źródłach. Zaproponowane algorytmy z powodzeniem zastosowałem także w projekcie headtrackera z równie dobrym rezultatem. Zawierają one w sobie element sterowania układem u_k, który eliminując ruchy liniowe, wpływa na lepszą stabilność heatrackera. Można to porównać



Rysunek 14. Filtrowanie odczytów z akcelerometru, Kalman vs Średnia

do przedstawionego wyżej filtrowania niskich prędkości obrotowych.

Generowanie ramki PPM

Zgodnie z opisem ramki PPM przedstawionym na początku, zakodowane kanały posiadają określoną długość, która pośrednio reprezentuje kąt obrotu kamery w samolocie. Pozostaje zamienić kąty **gyroAngleX** oraz **gyroAngleZ** na czasy wyrażone w µs:

```
int calcLenPPM(double gyroAngle,
double scale) {
  //Nowy czas kanału PPM
  int len = CH_LEN_
CNT + (gyroAngle*scale/
SERVO_MAX_ANLGE)*CH_LEN_CHG;

  //Sprawdzenie marginesów
  if (len < CH_LEN_MIN) return
CH_LEN_MIN;
  else if (len > CH_LEN_MAX)
return CH_LEN_MAX;
  else return len;
}
```

gdzie:

CH_LEN_CNT – czas kanału dla pozycji centralnej (1200μs),

CH_LEN_CHG – maksymalny czas zmiany długości kanału (512μs),

CH_LEN_MIN – minimalny czas kanału (1200μs-512μs),

CH_LEN_MAX – maksymalny czas kanału (1200μs+512μs),

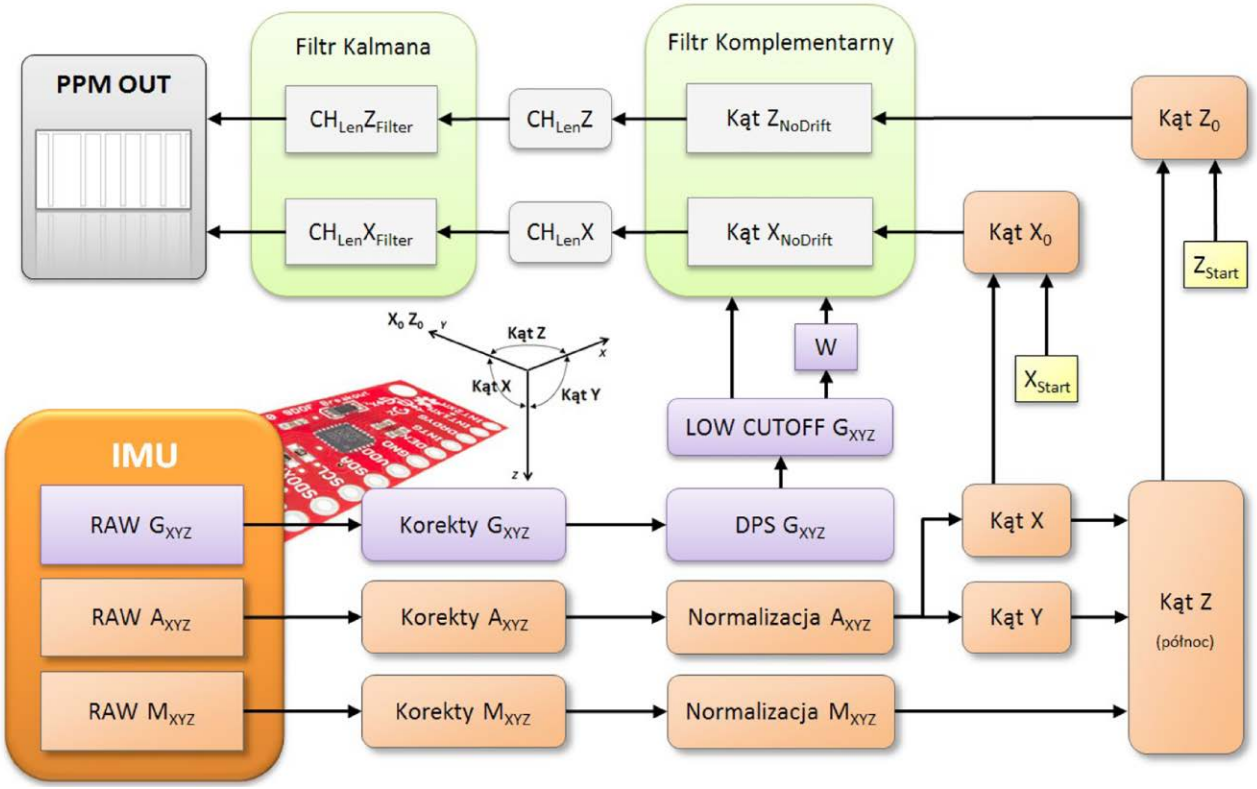
gyroAngle – kąt gyroAngleX i gyroAngleZ,

SERVO_MAX_ANLGE – dopuszczalny kąt wychylenia serwomechanizmu w stopniach (90°),

scale – współczynnik służący do sterowania czułością headtrackera, im wyższy tym szybszy obrót headtrackera.

Mamy już wszystkie informacje niezbędne do wygenerowania ramki PPM, która zostanie wysłana do nadajnika RC. Generowanie odbywa się w przerwaniu 16-bitowego timera, dzięki czemu jest niezakłócona przebiegiem pętli głównej:

```
ISR(TIMER1_COMPA_vect) {
  static uint16_t ch_len_isr[CH_
NUMBER];  //Czasy poszczególnych
kanałów PPM w przerwaniu
  static uint8_t ch_num=0;  //
Numer aktualnie generowanego
kanału w przerwaniu
  static uint8_t ch_dspl=0;  //0
- generacja przerwy pomiędzy
kanałami; 1 - generacja kanału
```



Rysunek 15. Algorytm wyznaczania kątów obrotu i sygnału PPM

```
  static uint16_t ch_fill=0;  //
Czas pozostały do wypełnienia
PPM_LEN

  if (ch_dspl==0) {
    OCR1A = PPM_DELAY;
    if (ch_num==0) {
      ch_fill = 0;
      //Przepisanie nowych
długości kanałów
      for (uint8_t
i=0; i<CH_NUMBER; i++)
ch_len_isr[i]=ch_len[i];
    }
    ch_dspl = 1;
    TCCR1A |= (1<<COM1A0); //
Następny stan wysoki
    ch_fill += PPM_DELAY;
  }
  else {
    if (ch_num==CH_NUMBER) {
      //Wypełnienie
      OCR1A = PPM_LEN-ch_fill;
      ch_num = 0;
    }
    else {
      //Jeden z kanałów
      OCR1A =
ch_len_isr[ch_num++];
      ch_fill +=
ch_len_isr[ch_num];
    }
    ch_dspl = 0;
    TCCR1A &=
~(1<<COM1A0); //Następny
stan niski
  }
}
```

Wykaz elementów

Wariant do montażu przewlekanego

Rezystory:
R1, R6: 240 Ω
R2: 390 Ω
R5: 10 kΩ

Kondensatory:
C1: 220 µF/16 V
C2: 47 µF/16 V
C3: 10 µF/16 V
C4: 100 µF/16 V
C5…C7: 100 nF
C8: 22 µF/16 V

Półprzewodniki:
IC1: moduł z LSM9DS0 (SparkFun)
IC2: LM317T lub odpowiednik
IC3: LM7805TV lub odpowiednik
IC4: ATmega8
LED: dioda LED 3 mm

Inne:
Podstawka 28pin
Przycisk
Odpowiednie listwy i gniazda goldpin

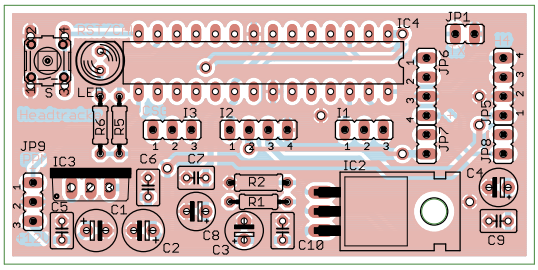
Wariant do montażu SMD

Rezystory: (SMD 0603)
R1…R4: 4,7 kΩ
R5: 10 kΩ
R6: 470 Ω

Kondensatory: (SMD 0603)
C1, C2: 100 nF
C3, C4: 22 pF

Półprzewodniki:
IC1: moduł z LSM9DS0 (SparkFun)
IC2: ATmega168 TQFP
LED: dioda LED SMD
T_SDA, T_SCL: BSS138

Inne:
16MHz: rezonator kwarcowy 16 MHz
Odpowiednie listwy i gniazda goldpin



Rysunek 16. Schemat montażowy HT w wersji przewlekanej

Sygnał PPM generowany jest na pinie procesora **OC1A** (**PB1**), który zmienia stan na przeciwny w każdym przerwaniu. W kodzie przerwania sterujemy rejestrem **OCR1A**, do którego zapisujemy czas opóźnienia do kolejnego przerwania. Zmiana stanu odbywa się poprzez sterowanie bitem **COM1A0** rejestru **TCCR1A**, dzięki któremu możliwe jest generowanie ramki o polaryzacji dodatniej lub ujemnej (PPM Positive lub PPM Negative). Za pomocą dodatkowej zmiennej rozpoznajemy czy w aktualnym przerwaniu będzie generowany kanał lub pauza. W każdym przerwaniu zapamiętujemy czasy poszczególnych kanałów po to, by w ostatnim przebiegu odpowiednio wysterować czas wypełnienia ramki (sygnał synchronizacji). Natomiast czasy kanałów są ustawiane w kodzie w ściśle określonym momencie, dzięki czemu nie mogą ulec zmianie w trakcie generowania ramki PPM. Aby dodatkowo upewnić się, że sygnał wysyłany do nadajnika jest spójny w każdym momencie, zmiany długości kanałów dokonujemy w bloku atomowym:

```
ATOMIC_BLOCK(ATOMIC_FORCEON) {
  ch_len[CH_ROLL] = newLenX;
  ch_len[CH_YAW] = newLenZ;
}
```

gdzie:

ch_len[] – tabela zawierająca długości kanałów,

CH_ROLL – numer kanału dla obrotu wokół osi X,

CH_YAW – numer kanału dla obrotu wokół osi Z,

*newLen** – nowe czasy kanałów.

Podsumowując, algorytm obliczania i filtrowania kątów został przedstawiony na **rysunku 15**.

Zerowanie położenia centralnego

Po uruchomieniu headtrackera następuje zapamiętanie aktualnych kątów kierunku i przechyłu IMU, względem których wykonywane są obroty kamery. Dzięki temu w sytuacji awaryjnej, na przykład po samoczynnym resecie procesora przez watchdog lub chwilowej utracie zasilania, kamera zawsze zostanie ustawiona w pozycji startowej (zostanie skierowana na dziób samolotu). Pozycja startowa zapisana w zmiennych programowych może zostać w każdej chwili skorygowana przez naciśnięcie przycisku. Okazuje się jednak, że po włączeniu zasilania, układ IMU zaczyna przesyłać poprawne dane z pewnym opóźnieniem. Z tego powodu reset położenia centralnego wykonywany jest w pętli przez określony czas regulowany za pomocą programowego timera.

```
if (BTN_PRESSED)
timer1=RESET_TIME;
(...)
if (timer1>0) {
  accXStart = accAngleX;
  magZStart = magAngleZ;
  gyroAngleX=gyroAngleZ = 0.0;
  newLenX = newLenZ = CH_LEN_CNT;
}
```

Po naciśnięciu przycisku zerowanie jest wielokrotnie powtarzany przez czas 200 ms. Okres ten jest wystarczająco długi, aby odczytać i obliczyć kąty z IMU i jednocześnie wystarczająco krótki, aby nie wprowadzać opóźnień. Wielokrotne powtarzanie wstępnych obliczeń jest także istotne ze względu na konieczność „wyżarzenia" filtru Kalmana, który w początkowych iteracjach zwraca wartości o niskiej jakości. Należy zwrócić uwagę, iż w procedurze resetującej nie następuje zerowanie zmiennych filtru Kalmana, dzięki czemu przejście od położenia poprzedniego do centralnego zostanie wykonane w sposób płynny. Operacja sygnalizowana jest przez zapalenie diody.

Montaż

Prototyp wykonano w wersji SMD i z przeznaczeniem dla mniej wprawnych osób do montażu przewlekanego. Dodatkowo, tę druga płytkę można wykonać samodzielnie, w warunkach domowych, na podstawie dokumentacji zawartej w materiałach dodatkowych. Schemat ideowy wariantu THT opublikowano w poprzednim odcinku (rys. 4), natomiast na **rysunku 16** pokazano jest schemat montażowy. Wśród użytych elementów znajdują się stabilizatory 7805T i 317T, kilka elementów filtrujących napięcie, przycisk, dioda świecąca, podstawka CPU (28 pinów) i gniazda goldpin. Zastosowano podciągnięcie wyprowadzenia *reset* poprzez rezystor 10 kΩ zapobiegające przypadkowemu restartowaniu się urządzenia. Zarówno wejścia, jak i wyjścia układów scalonych są filtrowane za pomocą par kondensatorów, elektrolitycznego i ceramicznego, co chroni przed skokami napięcia spowodowanymi

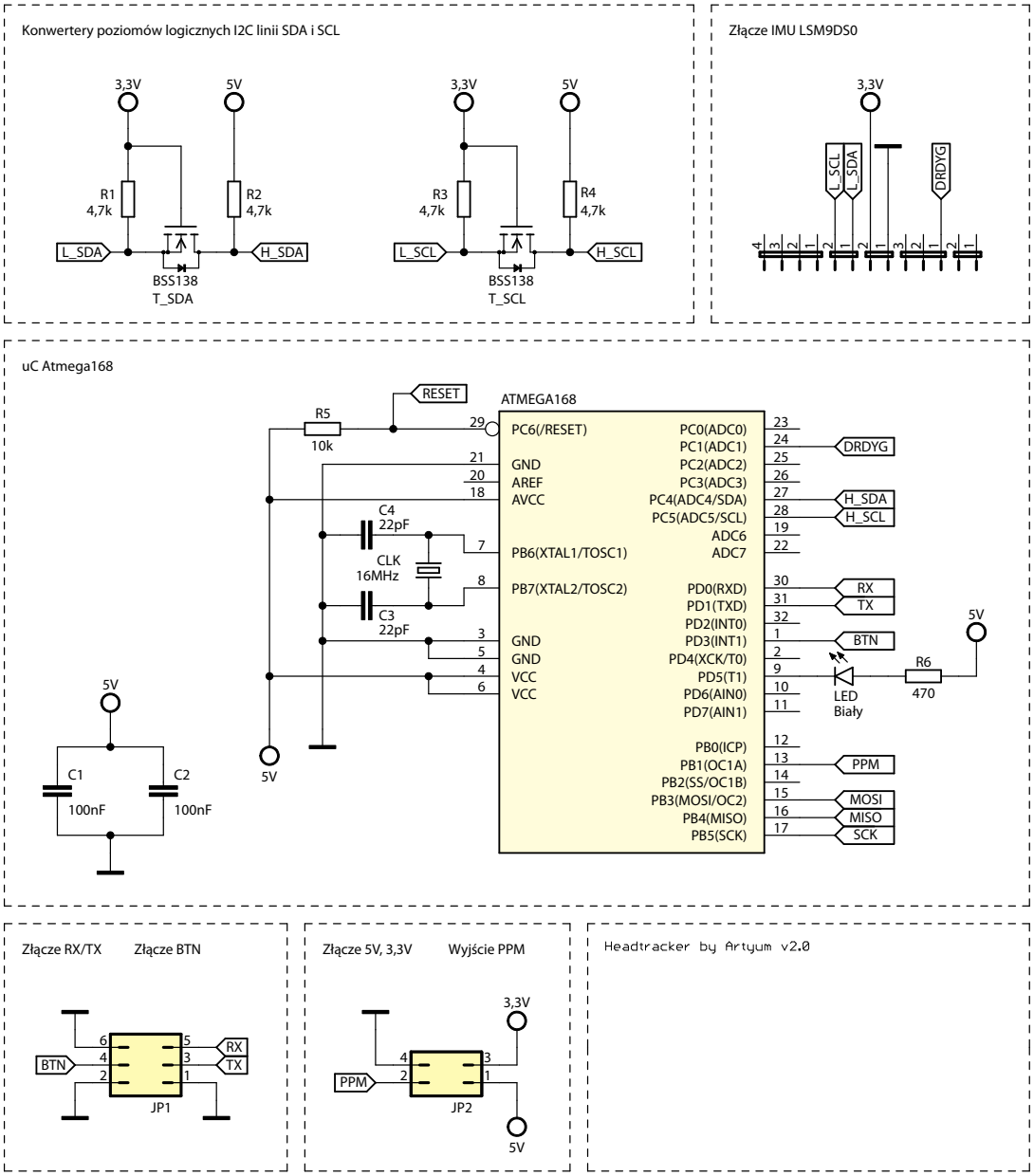
pracą CPU, zapewnia stabilną pracę układu, a także przyczynia się do ograniczenia emisji ciepła przez stabilizatory.

Przycisk jest wykorzystywany na dwa sposoby. Poprzez krótkie naciśnięcie jest zapamiętywane aktualne położenie IMU jako pozycja „centralna", względem której obliczane są względne kąty przechyłu i kierunku. Dłuższe przytrzymanie przycisku uruchamia procedurę kalibrowania żyroskopu.

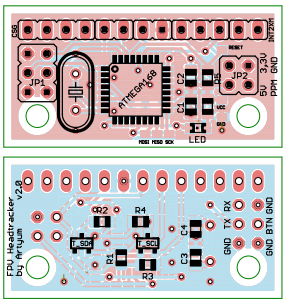
Headtracker może być zasilany napięciem z zakresu 7…25 V, a więc do jego pracy wystarczy bateria LiPo 2S lub 3S.

Na **rysunku 17** pokazano schemat ideowy HT w wersji SMD, a na **rysunku 18** jego schemat montażowy. Różnice pomiędzy wersją THT a SMD to między innymi zastosowanie Atmega168 w obudowie TQFP, zewnętrznego rezonatora 16 MHz oraz dwukrotne zmniejszenie wymiarów. W projekcie SMD założeniem było rozdzielenie funkcji zasilacza od modułu, stąd do uruchomienia i poprawnej pracy jest niezbędny zasilacz +5 V i +3,3 V. W tej wersji konwerter poziomów logicznych umieszczono na płytce. Zmianie uległ także przycisk – na płytce w wersji SMD utworzono piny umożliwiające dołączenie zewnętrznego przycisku, który może zostać umieszczony na w innym dogodnym miejscu.

W obu wersjach użyto tego samego układu IMU – modułu z LSM9DS0 firmy SparkFun. Płytka zawiera także pady niezbędne do przyłączenia programatora ISP oraz kondensatory filtrujące zasilanie.



Rysunek 17. Schemat ideowy HT w wersji SMD z mikrokontrolerem ATmega168



Rysunek 18. Schemat montażowy HT w wersji SMD z mikrokontrolerem ATmega168

Podsumowanie

Headtracker poprawnie działa już przy taktowaniu zegarem 8 MHz. Wersja oparta na mikrokontrolerze Atmega8 jest w pełni funkcjonalna, jednak ze względu na ograniczoną wielkość pamięć Flash trudno jest oprogramować dodatkową funkcjonalność. To ograniczenie staje się nieaktualne przy zastosowaniu Atmega168 wykorzystanego w wersji SMD. W tej wersji program może zostać rozbudowany o możliwość półautomatycznej kalibracji poprzez terminal PC, natomiast taktowanie 16 MHz sprawia, iż system działa z większą dokładnością. Płynność ruchów kamery jest zapewniona dzięki wykorzystaniu flirtowania Kalmana. Naturalny dryf żyroskopowy został zniwelowany prostym, ale bardzo skutecznym filtrowaniem komplementarnym. W połączeniu z dodatkowym algorytmem eliminującym drgania akcelerometru przy niskich prędkościach obrotowych żyroskopu uzyskujemy stabilny i wygodny w użytkowaniu system. Pozostaje zająć miejsce na „fotelu pilota" i podziwiać widoki.

Arkadiusz Witczak
arwi@go2.pl

Bibliografia:

https://goo.gl/ChsnSG
https://goo.gl/avDkl6
http://goo.gl/tfvgqc
http://goo.gl/p6GwqA
http://goo.gl/E3mbGO
http://goo.gl/sJXwCX
http://goo.gl/zBkmqU
http://goo.gl/OHYNPm
http://goo.gl/1ESj4T
http://goo.gl/GC2VdD
http://goo.gl/NVX50M
http://goo.gl/LYUDiO