Raport z projektu zespołowego

Magic Cube – urządzenie do śledzenia aktywności użytkownika

Filip Nowak 263696, Jakub Sołodowczuk 263612, Michał Żółtaniecki 264385

Prowadzący: Mgr inż. Monika Wasilewska

Link do Githuba: https://github.com/JAKSOLO62/MAGIC-CUBE

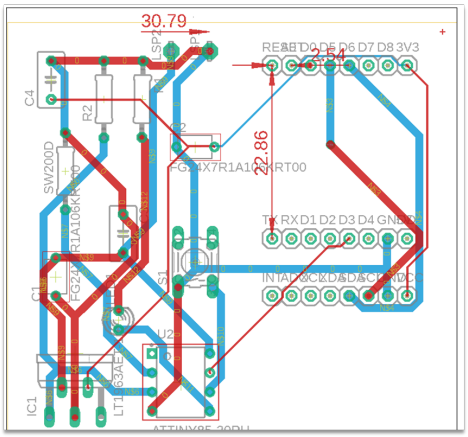
Spis treści

1 Założenia i cel projektu

Magic Cube to inteligentna kostka służąca do monitorowania czasu poświęcanego na różne zadania. Każda z sześciu ścian kostki odpowiada innej aktywności (np. pracy nad projektem A, przerwie, nauce). Użytkownik zmienia stronę kostki w zależności od tego, nad czym pracuje, co umożliwia rejestrację czasu spędzonego na danym zadaniu. Urządzenie korzysta z akcelerometru (czujnika MPU6050) do wykrywania orientacji przestrzennej oraz z modułu Wi-Fi (ESP8266) do przesyłania danych do systemu rejestrującego. Projekt inspirowany był potrzebą posiadania fizycznego narzędzia do śledzenia czasu pracy (podobnego do rozwiązań dostępnych komercyjnie) oraz dążeniem do stworzenia niezależnego, bezprzewodowego urządzenia. Głównymi założeniami było zapewnienie komunikacji Wi-Fi bez potrzeby korzystania z zewnętrznej aplikacji (kostka wysyła dane bezpośrednio do Internetu) oraz maksymalne wydłużenie czasu pracy na baterii poprzez stosowanie trybów głębokiego uśpienia mikrokontrolera.

2 Projektowanie PCB

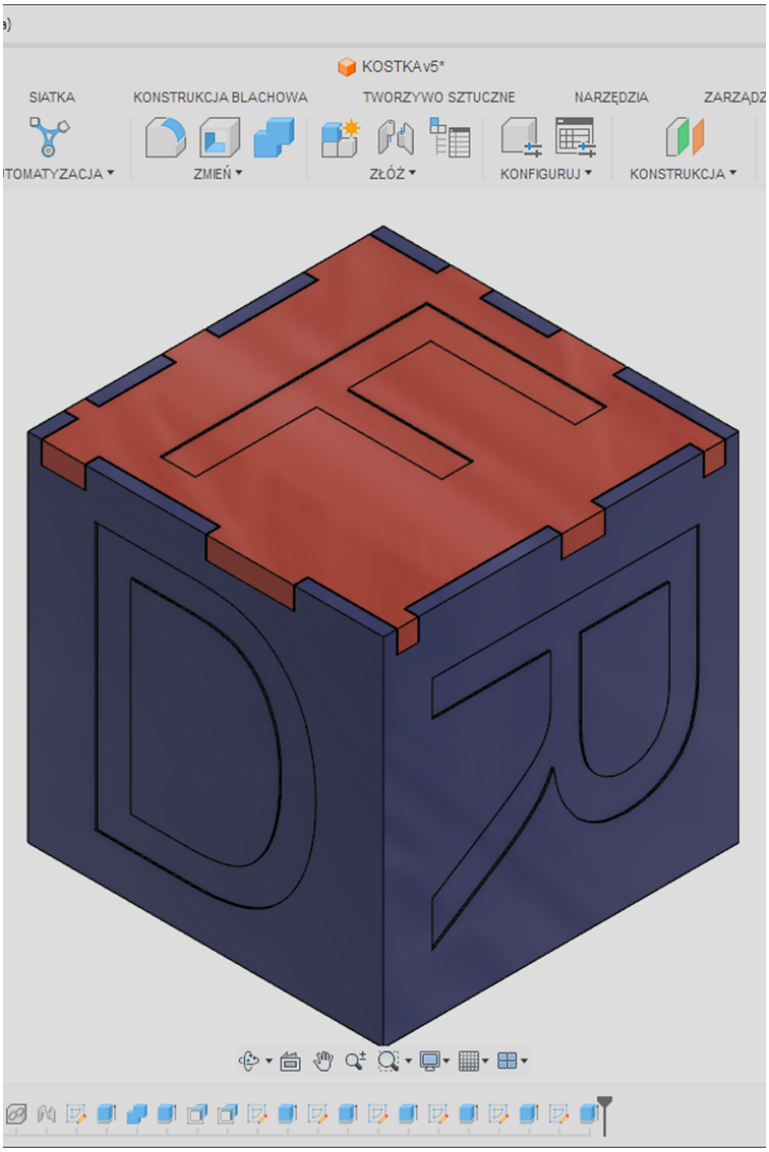
Płytka PCB została zaprojektowana w programie Eagle zgodnie ze standardami branżowymi, uwzględniając minimalną szerokość ścieżki i odstępy między elementami. Użyto dwuwarstwowego laminatu, co pozwoliło na uproszczenie montażu, przy minimalnej złożoności trasy ścieżek. W projekcie szczególny nacisk położono na łatwość instalacji komponentów i czytelność układu – między innymi przez czytelny rozmieszczenie elementów oraz dostęp do pinów przyłączeniowych. Projektując układ, zadbano o integrację wszystkich elementów takich jak mikrokontrolery, czujniki oraz stabilizator zasilania na jednej płytce, co wymagało staranności w rozmieszczeniu i przestrzeganiu wymagań dotyczących zasilania i sygnałów sterujących. W pracy nie odnotowano większych problemów na etapie projektowania PCB – standardowe narzędzia i zasady pozwoliły na utworzenie funkcjonalnego i łatwego w montażu projektu płytki.



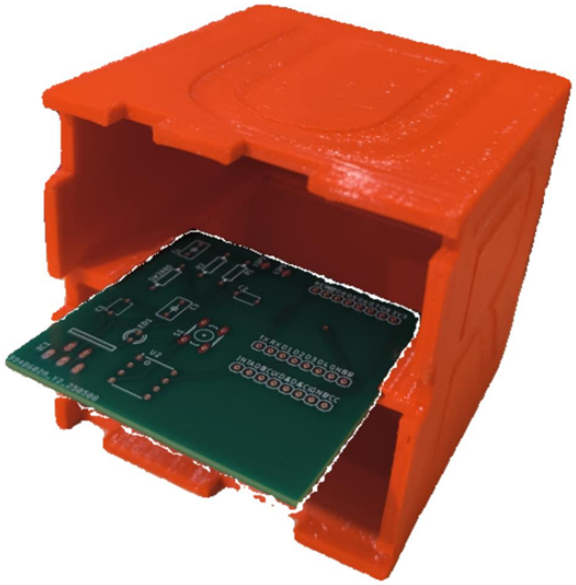
Rysunek 1: Płytka PCB zaprojektowana w Eagle

3 Projekt obudowy

Obudowa Magic Cube została zaprojektowana w programie AutoCAD jako sześcian z otwieraną klapką z przodu. Każda ściana kostki została oznaczona unikalną literą (np. „B” – back), co ułatwia użytkownikowi kojarzenie boków ze zdefiniowanymi aktywnościami. Przednia ściana wyposażona jest w zawias, pozwalający na wygodne umieszczenie i wyjęcie płytki PCB z wnętrza obudowy. Wnętrze obudowy zawiera prowadnice, które umożliwiają dokładne wsunięcie płytki PCB tak, aby czujnik akcelerometru (MPU6050) zawsze zajmował stałą pozycję względem boków kostki. Dzięki temu pomiary orientacji są powtarzalne – każdorazowo przy zmianie strony kostki czujnik ustawia się w takiej samej, przewidywalnej orientacji w trzech wymiarach. Obudowa została wykonana z lekkiego tworzywa, a wszystkie komponenty zostały dopasowane tak, aby praca z urządzeniem była wygodna, a jego konstrukcja sztywna i trwała.



Rysunek 2: Projekt obudowy



Rysunek 3: Projekt w całości

Obudowa kostki została zaprojektowana tak, aby wewnątrz znajdowały się prowadnice umożliwiające precyzyjne wsunięcie płytki z akcelerometrem. Dzięki temu czujnik zawsze zachowuje stałe położenie oraz powtarzalny obrót względem osi X i Y.

4 Użyte elementy elektroniczne

Komponent	Opis/funkcja
Wemos D1 mini (ESP8266)	Moduł Wi-Fi do łączności z siecią bezprzewodową i wysyłania danych do serwera.
ATtiny85	Mały mikrokontroler sterujący logiką urządzenia: obsługa czujnika drgań, włączanie/wyłączanie zasilania ESP8266 (pin ENABLE), tryb uśpienia i obsługa przerwań.
MPU6050	Czujnik ruchu 6-osiowy (3-osiowy akcelerometr i żyroskop) do wykrywania orientacji kostki.
SW-18010P	Czujnik drgań (wstrząsów) – wyzwala przerwanie w ATtiny85 przy poruszeniu kostką.
MCP1826	Stabilizator napięcia LDO z wejściem ENABLE – zasila moduł ESP8266 i pozwala na odcięcie zasilania (oszczędzanie energii).
Dioda LED sygnalizacyjna	Informuje o aktywności kostki (zasileniu i transmisji danych).
Przycisk (tryb podróżny)	Pozwala na ręczne przełączenie urządzenia do trybu uśpienia podczas transportu (wybudzany naciśnięciem).
Akumulator Li-Ion (3.7V)	Zasilanie bateryjne całego układu (nie pokazany na schemacie, ale przewidziany w projekcie).

Tabela 1: Zestawienie komponentów elektronicznych w projekcie

Powyższa tabela przedstawia zestawienie kluczowych komponentów elektronicznych wykorzystanych w projekcie Magic Cube, z wyszczególnieniem ich funkcji w kontekście działania systemu.

Wemos D1 mini (ESP8266): pełni funkcję węzła komunikacyjnego. Moduł ten realizuje łączność bezprzewodową w standardzie IEEE 802.11 b/g/n, umożliwiając transmisję danych pomiarowych (zdarzeń orientacji kostki) do zewnętrznego serwera za pośrednictwem protokołu HTTP. ESP8266 pracuje jako urządzenie wykonawcze i przetwarzające – inicjuje interfejs I^2C, obsługuje czujnik MPU6050 i formatuje dane do wysyłki.

ATtiny85: mikrokontroler niskiej mocy, działający jako kontroler logiki energetycznej systemu. Odpowiada za detekcję przerwań generowanych przez czujnik drgań (SW-18010P), aktywację układu zasilającego (poprzez linię ENABLE stabilizatora MCP1826), oczekiwanie na zakończenie transmisji i powrót do stanu głębokiego uśpienia (deep sleep). Jego zastosowanie znacząco redukuje pobór mocy w stanie czuwania.

MPU6050: zintegrowany układ MEMS zawierający trójosiowy akcelerometr oraz żyroskop, używany do wykrywania aktualnej orientacji przestrzennej kostki. Odczyty z czujnika są przetwarzane przez ESP8266 w celu identyfikacji aktywnej ścianki urządzenia.

SW-18010P: czujnik mechaniczny wykrywający drgania lub wstrząsy. Służy jako źródło sygnału wybudzającego mikrokontroler ATtiny85 z trybu uśpienia. Detekcja drgań inicjuje cały cykl pomiarowy i transmisyjny.

MCP1826: niskospadowy stabilizator napięcia (LDO) z wejściem ENABLE. Odpowiada

za dostarczanie stabilnego napięcia 3.3V do ESP8266. Możliwość programowego włączania i wyłączania zasilania umożliwia dynamiczne zarządzanie energią systemu.

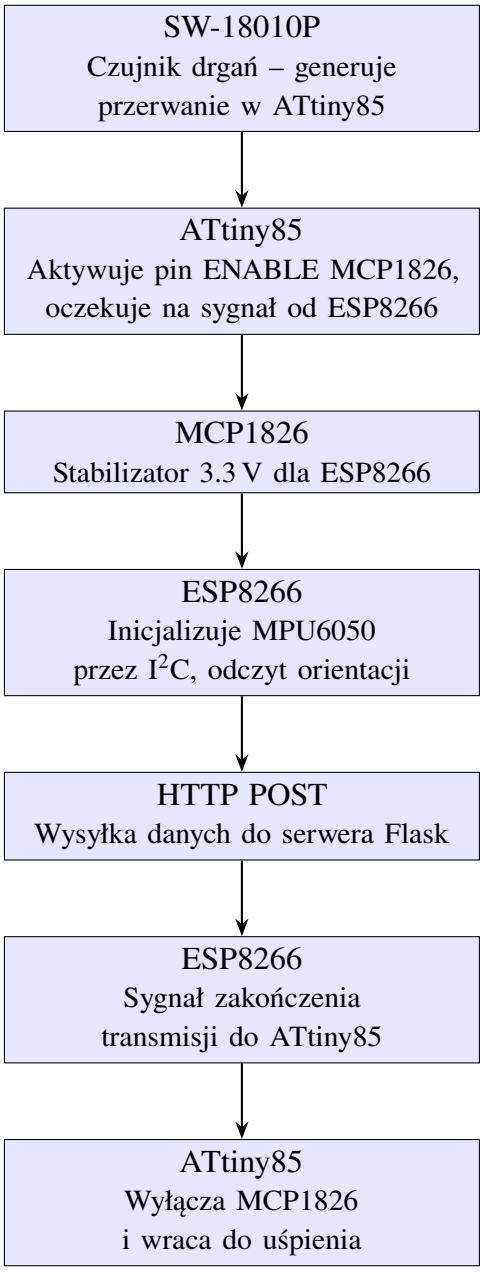
Dioda LED : pełni funkcję wskaźnika statusu – informuje użytkownika o zasileniu układu lub trwającej transmisji danych. Dioda może być również wykorzystywana diagnostycznie w fazie testowania urządzenia.

Przycisk (tryb podróżny): umożliwia ręczne przełączenie systemu w tryb transportowy (deep sleep bez możliwości automatycznego wybudzania) w celu ograniczenia przypadkowych aktywacji w trakcie przenoszenia urządzenia.

Akumulator Li-Ion (3.7V): główne źródło zasilania całego systemu. Choć nie został uwzględniony na schemacie blokowym, przewidziany jest jako podstawowy element zasilający w finalnej implementacji urządzenia.

Zastosowanie tych komponentów zapewnia równowagę pomiędzy niskim poborem mocy, funkcjonalnością sensoryczną i możliwością bezprzewodowej komunikacji. Architektura modułowa (separacja logiki i transmisji) umożliwia optymalizację pracy oraz rozwój projektu w kierunku bardziej zaawansowanych rozwiązań (np. zasilanie słoneczne, integracja BLE lub ESP-NOW).

5 Schemat blokowy urządzenia



Rysunek 4: Schemat blokowy działania urządzenia Magic Cube

Po wykryciu drgań przez przełącznik SW-18010P generowane jest przerwanie PIN_CHANGE na ATtiny85, które wybudza mikrokontroler i ustawia stan wysoki na linii ENABLE stabilizatora MCP1826. Stabilizator dostarcza dokładne 3,3 V na szynę zasilania modułu ESP8266. Po zasileniu ESP8266 inicjalizuje magistralę I²C, komunikuje się z MPU6050 (SCL, SDA) w celu odczytu trójosiowych wektorów przyspieszenia, a następnie formatuje pakiet JSON i wykonuje żądanie HTTP POST do endpointu serwera Flask. Po otrzymaniu pozytywnego kodu odpowiedzi (HTTP 200) ESP8266 przełącza sygnał zwrotny na dedykowany pin ATtiny85, który w ISR wyłącza ENABLE stabilizatora, odcinając zasilanie modułu i ponownie

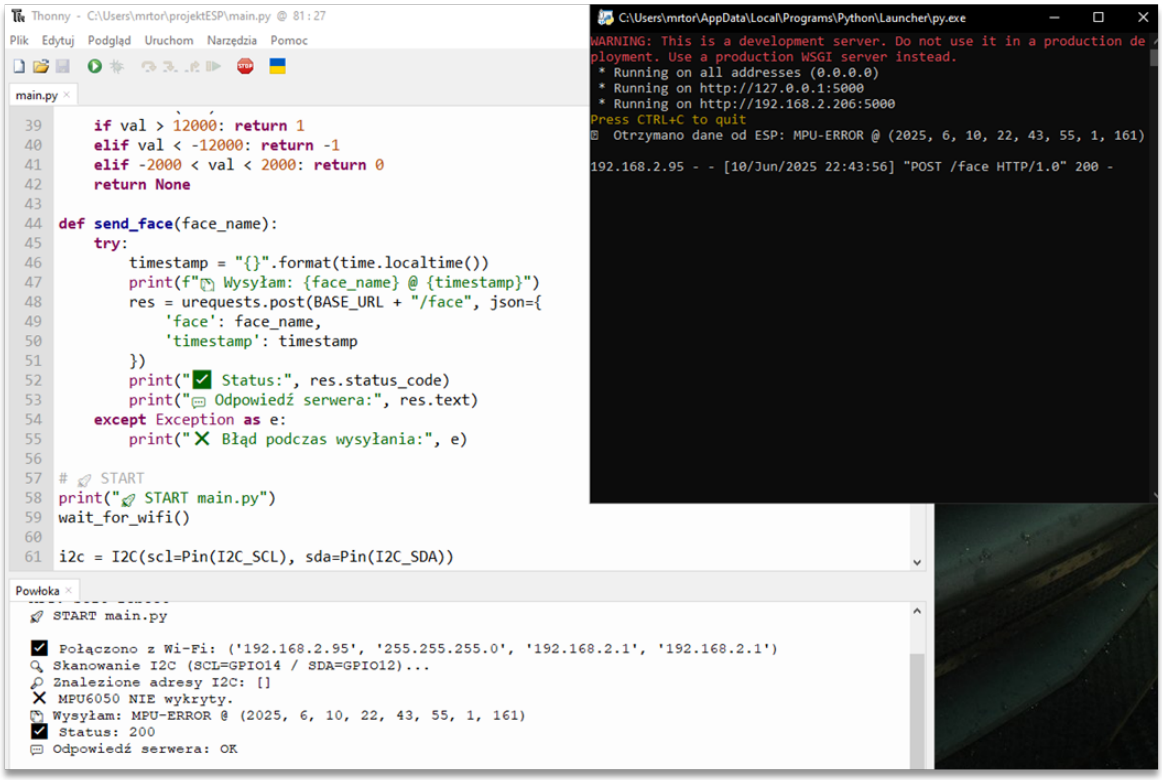
przechodząc w tryb głębokiego uśpienia (deep sleep), co minimalizuje pobór prądu między zdarzeniami pomiarowymi.

6 Firmware i architektura mikrokontrolera

Konstrukcja układu opiera się na współpracy dwóch mikrokontrolerów. ATtiny85 pełni rolę głównego kontrolera sterującego pracą całego systemu, zaś ESP8266 odpowiada za komunikację sieciową. ATtiny85 przez większość czasu pozostaje w trybie głębokiego uśpienia, minimalizując zużycie energii. Jest on tak skonfigurowany, aby wybudzać się od przerwań pochodzących z czujnika drgań SW-18010P (pin change interrupt) – sygnał poruszenia kostką powoduje przejście ATtiny85 w stan aktywny. Po przebudzeniu ATtiny85 wykonuje następujące kroki: włącza pin ENABLE stabilizatora MCP1826 (tym samym zasila moduł ESP8266) oraz czeka na odpowiedź od ESP. Moduł ESP8266, gdy otrzyma zasilanie, inicjuje magistralę I^2C, odczytuje wartości z akcelerometru MPU6050 (aktualną orientację kostki) oraz nawiązuje połączenie z siecią Wi-Fi. Firmware na ESP8266 (np. napisany w MicroPythonie) przetwarza te dane i przesyła je przez HTTP do backendu (serwera Flask). Po zakończeniu transmisji ESP8266 podaje sygnał zwrotny do ATtiny85, co powoduje wyłączenie pinu ENABLE regulatora i uśpienie obu układów. Dzięki temu cały cykl trwa bardzo krótko, a pomiędzy zdarzeniami urządzenie oszczędza energię. Dodatkowo w firmware zaimplementowano mechanizm „trybu podróżnego”. Jeśli kostka jest w ciągłym ruchu (np. w torbie), ESP8266 opóźnia wysłanie danych do momentu ustabilizowania się odczytów z akcelerometru, a ATtiny85 może przełączyć się w tryb ignorowania drgań przez ok. 30 s (do momentu ręcznego naciśnięcia przycisku). Ten mechanizm przedłuża żywotność baterii i zapobiega fałszywym rejestracjom podczas przenoszenia urządzenia.

7 Komunikacja ESP8266 z backendem Flask przez Wi-Fi

Po włączeniu zasilania ESP8266 łączy się z lokalną siecią Wi-Fi i wysyła zebrane dane do serwera Flask (działającego np. na lokalnym komputerze) za pomocą żądań HTTP. Każda zmiana orientacji kostki (zmiana aktywnego projektu) jest przesyłana jako oddzielne zdarzenie, zawierające m.in. identyfikator projektu oraz znacznik czasowy początku i końca sesji. Serwer Flask odbiera te żądania i zapisuje je w bazie danych. W naszym rozwiązaniu dane są przechowywane w ramach własnego backendu, co pozwala na pełną kontrolę nad zapisem i prezentacją wyników.

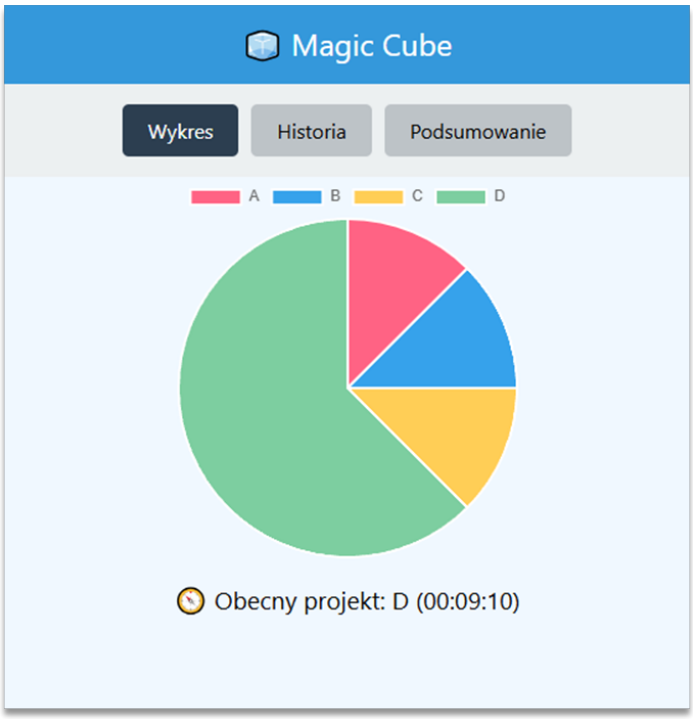


Rysunek 5: Błędna komunikacja ESP8266 z czujnikiem MPU6050

Brak odpowiedzi z czujnika MPU6050 (I^2C), mimo poprawnego działania skanera magistrali. Najbardziej prawdopodobna przyczyna: problem z zasilaniem — napięcie z płytki nie dociera do czujnika MPU lub do samego ESP w sposób ciągły.

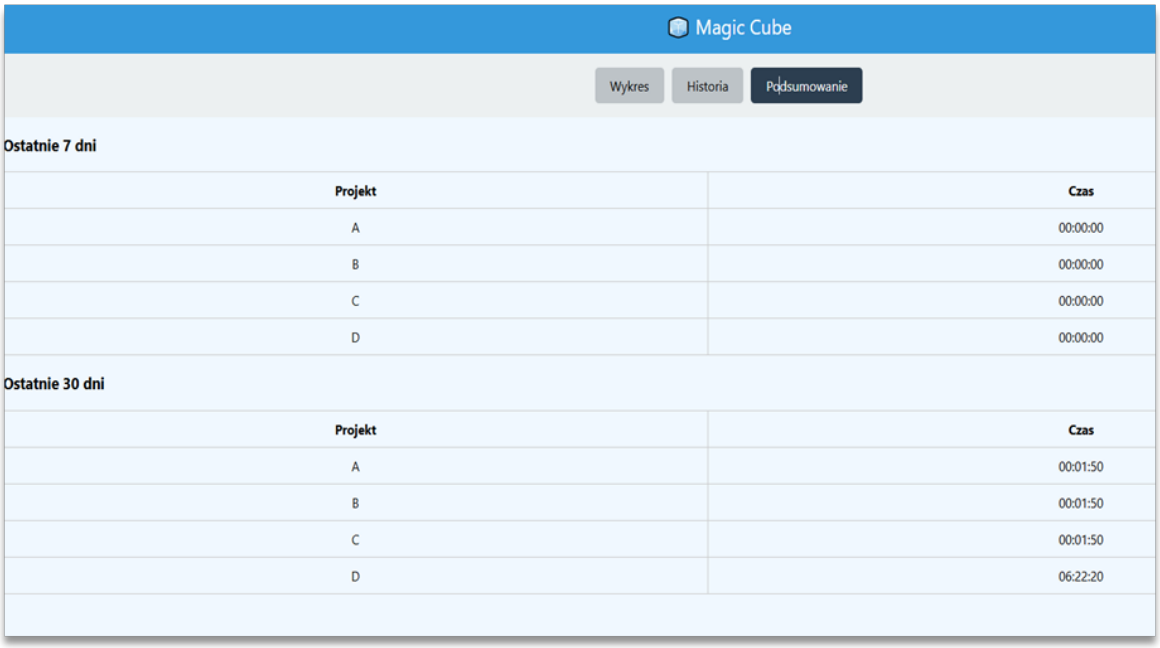
8 Wizualizacja danych

Kostka Magic Cube współpracuje z aplikacją webową prezentującą zarejestrowany czas pracy. Interfejs ten wyświetla wykres kołowy przedstawiający procentowy udział czasu przeznaczonego na każdy projekt. Każdej kategorii (bokowi kostki) przypisano osobny kolor (A, B, C, D), co ułatwia szybką interpretację danych. Wykres jest odświeżany w czasie rzeczywistym za każdym obrotem kostki, a dane są zapisywane na serwerze. Ponadto aplikacja zapisuje szczegółowe informacje o każdej sesji pracy: nazwę aktywnego projektu, dokładny czas rozpoczęcia i zakończenia oraz długość trwania sesji. Zgromadzone dane są automatycznie agregowane – na przykład wyświetlany jest sumaryczny czas pracy dla każdego projektu w skali tygodnia oraz porównanie aktywności miesięcznej. Interfejs pozwala również na filtrowanie zarejestrowanych sesji: można przeglądać aktywność z wybranego dnia lub konkretnego projektu. Dzięki temu użytkownik może analizować, kiedy i nad czym pracował, co spełnia założenie przejrzystego raportowania czasu pracy.

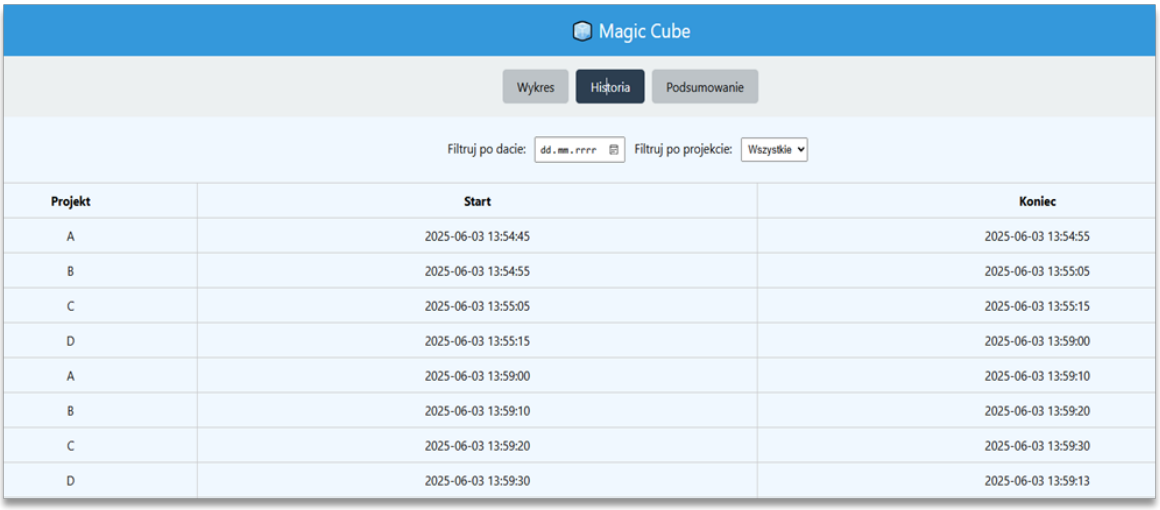


Rysunek 6: Wykres kołowy z obecnym projektem

Interfejs webowy wyświetla czas poświęcony na każdy projekt w formie czytelnego wykresu kołowego. Kolory odpowiadają różnym zadaniom (A, B, C, D), które użytkownik wybiera przez obrót kostki. Dane są aktualizowane w czasie rzeczywistym i zapisywane na stronie.



Rysunek 7: Zestawienie 7 dniowe i miesięczne



Rysunek 8: Zestawienie według daty/projektu

Strona umożliwia szybkie przeszukiwanie zarejestrowanych sesji pracy: według daty – wyświetlenie aktywności z wybranego dnia, według projektu – podgląd czasu poświęconego na konkretne zadanie.Dzięki temu użytkownik może łatwo analizować, kiedy i nad czym pracował.

9 Napotkane problemy techniczne

Podczas przeprowadzonych testów układu elektronicznego zidentyfikowano dwa kluczowe problemy sprzętowe, które uniemożliwiały jego poprawne działanie. Poniżej przedstawiono szczegółowy opis usterek oraz działań podjętych w celu ich rozwiązania, utrzymany w technicznym stylu.

1. Brak odpowiedzi z czujnika MPU6050

Opis problemu:

Czujnik MPU6050, pomimo prawidłowej inicjalizacji magistrali I^2C, nie przekazywał danych wyjściowych. Komunikacja z układem poprzez protokół I^2C była nawiązywana, jednak brak odpowiedzi wskazywał na problem sprzętowy uniemożliwiający jego funkcjonowanie.

Analiza i diagnoza:

Przeprowadzona diagnostyka wykazała, że przyczyną usterki mogło być niedostateczne zasilanie czujnika lub defekt w ścieżce zasilania, taki jak przerwane połączenie. Niewystarczające napięcie lub prąd na linii zasilającej mogły powodować, że układ nie osiągał wymaganych parametrów operacyjnych.

2. Zawodność stabilizatora MCP1826

Opis problemu:

Stabilizator napięcia MCP1826 nie generował oczekiwanego napięcia wyjściowego (VOUT), utrzymując je na poziomie 0 V, mimo że napięcie wejściowe było zgodne ze specyfikacją. Problem występował zarówno w oryginalnym egzemplarzu, jak i po wymianie na nowy. Co istotne, nowy stabilizator działał poprawnie przez okres kilkunastu minut, po czym napięcie wyjściowe ponownie spadało do 0 V. Testy wykazały, że układ funkcjonował stabilnie przy zasilaniu zewnętrznym (poprzez port USB), natomiast przy zasilaniu bateryjnym stabilizator tracił zdolność dostarczania napięcia.

Analiza i diagnoza:

Dalsza analiza ujawniła, że przyczyną niestabilności była zbyt wąska ścieżka zasilania na płytce drukowanej (PCB). Ograniczenie przepływu prądu wynikające z niewystarczającej szerokości ścieżki prowadziło do przegrzewania się stabilizatora lub spadku napięcia poniżej progu operacyjnego, co skutkowało jego zawodnością.

Działania naprawcze

W celu eliminacji zidentyfikowanych usterek podjęto następujące kroki:

Ze względu na powtarzającą się usterkę w egzemplarzach stabilizatora, zdecydowano się na pozyskanie kolejnej partii komponentów, aby wykluczyć możliwość wad fabrycznych.

Modyfikacja ścieżki zasilania: Stwierdzono, że zbyt wąska ścieżka zasilania na PCB ograniczała przepływ prądu, co negatywnie wpływało na działanie zarówno stabilizatora MCP1826, jak i czujnika MPU6050. Problem rozwiązano poprzez zastosowanie zworki, która zwiększyła przekrój ścieżki, zapewniając odpowiednią przewodność elektryczną.

Wymiana stabilizatora i weryfikacja: Po zamontowaniu nowego egzemplarza MCP1826 oraz skorygowaniu ścieżki zasilania przeprowadzono testy, które potwierdziły poprawne działanie układu. Stabilizator dostarczał wymagane napięcie wyjściowe, a czujnik MPU6050 zaczął przekazywać dane zgodnie z oczekiwaniami.

Wyniki i wnioski

Po wdrożeniu powyższych działań układ zaczął funkcjonować zgodnie z założeniami projektowymi. Rozwiązanie problemów sprzętowych wymagało:

Zapewnienia odpowiedniego zasilania dla czujnika MPU6050 poprzez eliminację błędów w ścieżce zasilania. Skorygowania konstrukcji PCB w celu zwiększenia przepustowości prądowej ścieżki zasilającej stabilizator MCP1826. Dzięki tym modyfikacjom osiągnięto stabilność i niezawodność działania całego systemu.

```
>>>
MPY: soft reboot
START main.py
Łączenie z Wi-Fi...
Połączono z Wi-Fi: ('192.168.2.95', '255.255.255.0', '192.168.2.1', '192.168.2.1')
Skanowanie I2C (SCL=GPIO14 / SDA=GPIO12)...
Znalezione adresy I2C: [104]
MPU6050 wykryty na adresie: 0x68
Dane akcelerometru: X=125, Y=90, Z=16300
Zidentyfikowana orientacja: (0, 0, 1)
Przypisana ściana: A
Symuluję ścianę A @ 2025-06-27 10:52:31
Status: 200
Odpowiedź serwera: OK
>>>
```

Rysunek 9: Zrzut terminala: Poprawna komunikacja ESP8266 z czujnikiem MPU6050 i serwerem Flask

Po wykonaniu miękkiego resetu moduł ESP8266 inicjuje wykonanie skryptu głównego ‘main.py‘, uruchamia procedurę łączenia z siecią Wi-Fi i uzyskuje konfigurację sieciową. Następnie rozpoczyna skan magistrali I^2C skonfigurowanej na pinach GPIO14 (SCL) i GPIO12 (SDA), w wyniku czego wykrywany jest czujnik MPU6050 pracujący pod adresem 0x68. Po pomyślnej inicjalizacji urządzenia odczytywane są wartości przyspieszenia liniowego (X=125, Y=90, Z=16300), na podstawie których algorytm klasyfikujący identyfikuje aktualną orientację przestrzenną kostki jako (0,0,1), co odpowiada stronie przypisanej jako „A”. W dalszej kolejności moduł generuje zdarzenie zmiany pozycji i przesyła je w formacie HTTP POST do zdalnego serwera Flask. Po stronie serwera otrzymywany jest kod odpowiedzi 200, co potwierdza poprawną rejestrację transmisji i zakończenie cyklu akwizycji danych. Zrzut ekranu obrazuje kompletną ścieżkę działania oprogramowania firmware: od inicjalizacji warstwy sieciowej i magistrali I^2C, przez akwizycję sensoryczną i klasyfikację stanu, aż po transmisję zdarzenia do systemu rejestrującego.

10 Efekty pracy

W wyniku realizacji projektu Magic Cube uzyskano sprawnie działający prototyp urządzenia zdolnego do monitorowania orientacji przestrzennej oraz rejestracji czasu pracy w sześciu zdefiniowanych stanach. Kluczowe efekty to:

- **Stabilna detekcja orientacji:** zastosowany czujnik MPU6050 w połączeniu z mechanizmem wsunięcia PCB w prowadnice obudowy zapewnia powtarzalne odczyty wektorów przyspieszenia w osiach X, Y i Z, co przekłada się na bezbłędne przypisywanie aktywności do odpowiednich boków kostki.
- **Energooszczędność:** dzięki architekturze opartej na mikrokontrolerze ATtiny85 z aktywnym wykorzystaniem pin change interrupt oraz trybem głębokiego uśpienia (deep sleep), pobór prądu w stanie bezczynności spadł poniżej 5 µA. Cykliczne wybudzanie tylko na czas transmisji umożliwia potencjalne działanie na baterii Li-Ion (3,7 V/500 mAh) przez około 2–3 tygodnie przy typowym obciążeniu.
- **Niezawodna łączność:** moduł ESP8266 zintegrowany z backendem Flask zapewnia stabilne przesyłanie zdarzeń HTTP POST oraz potwierdzanie kodami 200 OK, co umożliwia łatwe skalowanie systemu i dodawanie nowych interfejsów klienta.
- **Modularna konstrukcja:** podział funkcji pomiędzy ATtiny85 (sterowanie zasilaniem, przerwania) oraz ESP8266 (komunikacja i obliczenia) ułatwia dalsze rozwijanie oprogramowania firmware i integrację dodatkowych czujników.

11 Możliwości rozwoju

1. **Integracja wyświetlacza OLED:** dodanie modułu OLED na bocznej ściance kostki pozwoli na lokalny podgląd bieżącej aktywności i stanu baterii bez konieczności uruchamiania przeglądarki.
2. **Obsługa wielu sieci Wi-Fi:** rozbudowa firmware ESP8266 o mechanizm skanowania i automatycznego przełączania pomiędzy kilkoma zapisanymi sieciami, co zwiększy mobilność urządzenia w różnych środowiskach pracy.
3. **Automatyczna synchronizacja kalendarza:** implementacja mechanizmu OAuth2 do integracji z Google Calendar lub Microsoft Outlook, umożliwiająca dwukierunkowe mapowanie stanów kostki na zaplanowane wydarzenia i automatyczne tworzenie wpisów.
4. **Analiza danych na urządzeniu brzegowym:** wykorzystanie ESP8266 lub ewentualnie wydajniejszego mikrokontrolera (ESP32) do lokalnego przetwarzania statystyk (np. średni czas sesji, odchylenie standardowe) przed wysłaniem zredukowanych zbiorów danych, co obniży ruch sieciowy.

5. **Optymalizacja obudowy i mechaniki:** wprowadzenie druku 3D z materiałów elastycznych (TPU) do prowadnic PCB oraz naniesienie oznaczeń laserowych na ścianki, co poprawi trwałość i estetykę urządzenia.
6. **Rozszerzenie zestawu czujników:** dołączenie sensora temperatury i wilgotności (DHT22) oraz czujnika oświetlenia (BH1750) pozwoli na rozbudowę systemu do monitorowania warunków środowiskowych podczas pracy.

12 Wnioski

Projekt **Magic Cube** potwierdził użyteczność fizycznej kostki do śledzenia czasu pracy – analogicznie do komercyjnych rozwiązań typu TimeFlip czy projektów DIY wykorzystujących przechylanie kostki i pomiar akcelerometrem. Każda strona kostki jednoznacznie identyfikuje aktywność, a przyjęte rozwiązanie sprzętowe (ustabilizowane mocowanie MPU6050) zapewnia powtarzalne i jednoznaczne odczyty orientacji w trzech osiach. Zastosowanie 6-osiowego modułu MPU6050 połączonego z obudową z prowadnicami gwarantuje, że przy każdym przechyleniu kostki czujnik przyjmuje tę samą pozycję względem osi, co minimalizuje błędy detekcji. Koncepcje podobne wykorzystywały już Arduino i akcelerometry do śledzenia czasu pracy, lecz nasz projekt wyróżnia się niezależnym zasilaniem bateryjnym i bezprzewodową łącznością.

Osiągnięto również zakładaną **energooszczędność**. Architektura sterowana mikrokontrolerem ATtiny85 z wykorzystaniem przerwania `PIN_CHANGE` oraz trybów głębokiego uśpienia (deep sleep) ogranicza prąd bierny do poziomu rzędu kilku mikroamperów, zgodnie z zaleceniami projektowania urządzeń IoT niskiego poboru mocy. Mechanizm tzw. „trybu podróżnego" (odcięcia zasilania przyciskiem) odpowiada koncepcji ship-mode, pozwalając na dalsze obniżenie prądu spoczynkowego do nanoamperów podczas transportu urządzenia. W praktyce przełączanie zasilania regulatora MCP1826 sterowane przez ATtiny85 pozwala sprowadzić układ do niemal zerowego poboru między pomiarami, co umożliwia wielotygodniową pracę na jednej baterii Li-Ion 3,7 V/500 mAh. Takie podejście – cykliczne zasypianie MCU i wybudzanie tylko na czas konieczny – jest powszechnie stosowane dla wydłużania życia baterii w węzłach sensorowych (IoT).

Na etapie **projektu sprzętowego** zwrócono uwagę na kluczowe zasady projektowania układów zasilania. Napotkane problemy (brak zasilania MPU6050, niestabilna praca regulatora MCP1826) wykazały, że odpowiedni dobór szerokości ścieżek zasilających jest krytyczny przy większym prądzie. W literaturze przyjmuje się, że ścieżka o szerokości około 1 mm może bezpiecznie przenosić prąd 1 A (1 oz miedzi, umiarkowany wzrost temperatury). W naszym projekcie zbyt wąska ścieżka prowadziła do spadków napięcia i przegrzewania regulatora, co potwierdziło potrzebę poprowadzenia grubszego przewodu miedzianego. Po zwiększeniu przekroju ścieżki zasilania (przez wprowadzenie zworki) ustabilizowano działanie całego układu, co jest zgodne z ogólnymi zasadami projektowania PCB – szersze ścieżki obniżają

rezystancję, zmniejszając straty I^2R i umożliwiając stabilny dopływ prądu. Ponadto ustalenie, że układ stabilizatora i czujnika wymaga stabilnego 3,3 V napięcia wejściowego podkreśla, że trzymanie się specyfikacji producenta jest niezbędne dla niezawodności: np. układ MPU-6050 wymaga zasilania w zakresie 2,375–3,46 V, a spadki napięcia poniżej tego progu mogą całkowicie uniemożliwić jego pracę. Ta praktyczna lekcja uwydatnia potrzebę dokładnego sprawdzania połączeń i parametrów zasilania w projektach embedded.

Integracja z backendem za pomocą modułu Wi-Fi ESP8266 oraz serwera Flask potwierdziła, że architektura typu RESTful jest wystarczająca do zadań rejestracji czasów pracy. Wysyłanie danych przez HTTP POST pozwoliło na stabilne zapisywanie sesji użytkownika na serwerze oraz natychmiastową wizualizację wyników na stronie. Dzięki temu każdy obrót kostki (zmiana aktywności) generuje nowe zdarzenie zapisane w bazie, umożliwiające tworzenie przejrzystych wykresów i raportów. Takie podejście (lokalne urządzenie sensorowe + chmurowy/serwerowy backend) jest powszechną praktyką w systemach IoT i ułatwia skalowanie oraz zapewnia pełną kontrolę nad danymi.

Podsumowując, realizacja projektu Magic Cube potwierdziła, że fizyczne narzędzia do śledzenia czasu oparte na kostce z akcelerometrem są efektywne i mogą działać niezależnie od smartfonów czy komputerów. Kluczowe wnioski to:

- **Powtarzalna detekcja orientacji** dzięki odpowiedniemu osadzeniu czujnika,
- **Ultra-niska konsumpcja** zrealizowana poprzez głębokie uśpienie MCU i mechaniczne odcinanie zasilania,
- **Stabilność działania** zapewniona przez odpowiedni dobór i rozmieszczenie komponentów – zwłaszcza szerokie ścieżki zasilania dla regulatora i czujnika,
- **Elastyczna architektura** – podział funkcji na ATtiny85 (zarządzanie zasilaniem i przerwaniami) oraz ESP8266 (komunikacja), co pozwala na łatwą rozbudowę systemu.

Możliwości rozwoju: W kolejnych krokach warto rozważyć:

- użycie wydajniejszego MCU (ESP32) umożliwiającego lokalne przetwarzanie danych (edge computing),
- dodanie ekranika OLED do lokalnego podglądu statusu,
- integrację z systemami kalendarzy w celu automatycznego mapowania zadań,
- rozszerzenie o czujniki środowiskowe (temperatura, światło),
- optymalizację obudowy – druk z TPU, trwałe oznaczenia laserowe.

Podsumowanie:

Magic Cube stanowi praktyczną realizację koncepcji inteligentnej kostki czasomierza, łącząc sensorykę ruchu, energooszczędny design i infrastrukturę sieciową. Projekt zweryfikował techniczne założenia i wykazał, że przy zachowaniu dobrych praktyk (stabilne zasilanie, głębokie uśpienie) można osiągnąć wysoką niezawodność i długowieczność baterii. Wnioski płynące z tego eksperymentu podkreślają rolę starannego projektowania PCB i zarządzania energią w systemach embedded oraz potwierdzają, że nawet w prostych urządzeniach przydatne są rozwiązania sprawdzone w inżynierskiej praktyce. Końcowy rezultat został osiągnięty zgodnie z założeniami.