BMA400:

* Tryb pracy: low power – po wykryciu ruchu automatycznie przechodzi do normal.
* Zasada działania: czujnik reaguje na ruch w jakiejkolwiek osi, trzeba dostosować, żeby nie zbierał małych drgań, jak np. wiatr zawieje. Jak wykryje ruch, to wybudza STM
* Dane są przesyłane przez I2C do STM32

STM32:

* Tryb uśpienia: STANDBY – wybudzanie sygnałem z INT czujnika.
* Przeliczone procenty są przesyłane na bieżąco do ESP przez UART.
* Trzeba np. analizatorem stanów logicznych sprawdzić czy po wybudzaniu się nie powoduje jakichś skoków na PA5 (EN\_IO) i ew. skorygować w programie, bo to by bardzo dużo energii pobierało.

ESP8266:

* Zawsze wyłączony. Aktywowany przez STM32 na pinie EN\_ESP podczas transmisji.
* Przesyła procent otwarcia do drugiego ESP przez ESP-NOW.

Zasada działania:

* BMA400 w trybie low-power monitoruje wszystkie osie.
* Wykrycie ruchu ⇒ wake-up interrupt ⇒ przejście w tryb normal ⇒ INT1 = LOW.
* STM32 wybudzany z STANDBY przez pin PWR\_WAKEUP\_PIN1.
* STM32 sprawdza próbki z FIFO. Oblicza odchylenie standardowe z ostatnich 5 próbek, potem z kolejnych 5 i z kolejnych 5. Jeśli różnica między kolejnymi obliczonymi odchyleniami jest większa niż 5 g, to znak, że brama się rusza. STM przy każdym wybudzeniu tak oblicza, czyli przez cały czas trwania ruchu. Jeśli różnica będzie mniejsza niż 5 g to znak, że brama stoi. Próbki są zbierane co 10 ms.
* STM32 co 2 s aktywuje przerwanie timera (dopóki nie przekroczy time\_down/time\_up):
  + Włącza ESP (PA5 = HIGH), czeka 200 ms.
  + Wysyła aktualny czas przez UART do ESP.
  + ESP wysyła dalej do drugiego ESP (chyba)
  + Wyłącza ESP (PA5 = LOW).
  + Wchodzi ponownie w STANDBY.
* Brak timeoutu na przesłanie danych – dopóki nie zajdzie transmisja, to reszta kodu się nie wykona (87 us na ramkę 8-bit)