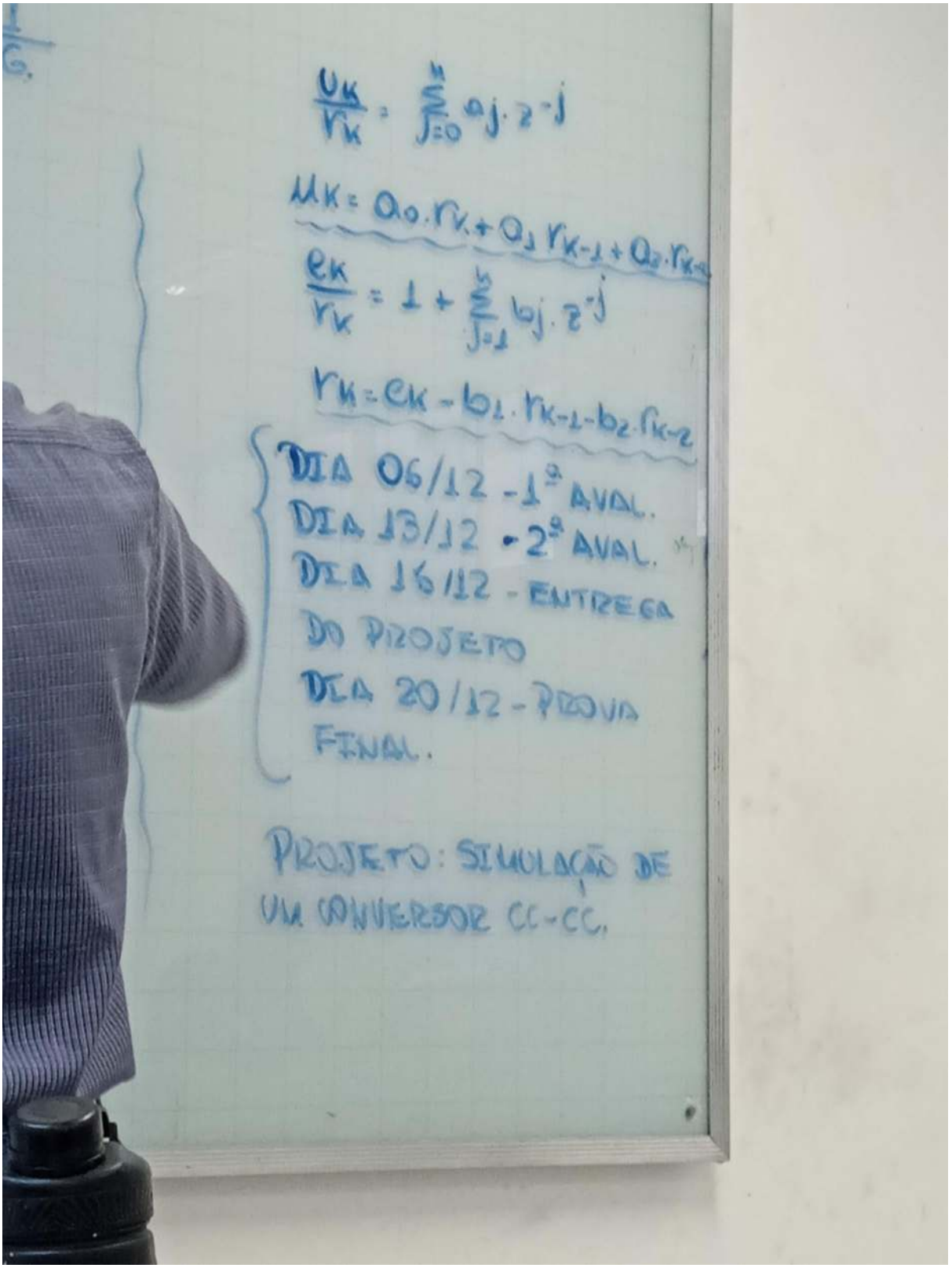
Cristiano Coutinho Costa

29/11

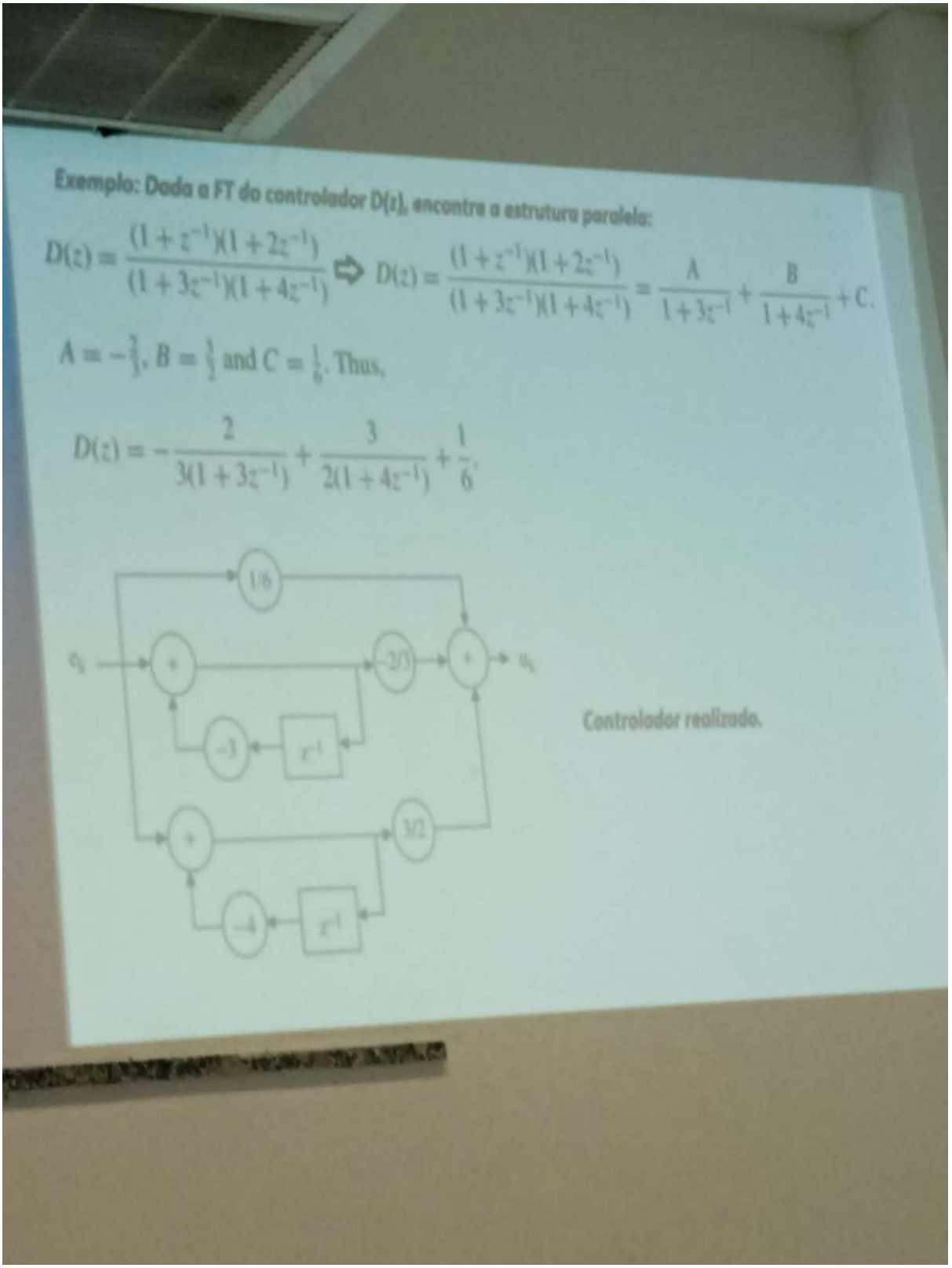
Prova



Controladores Digitais

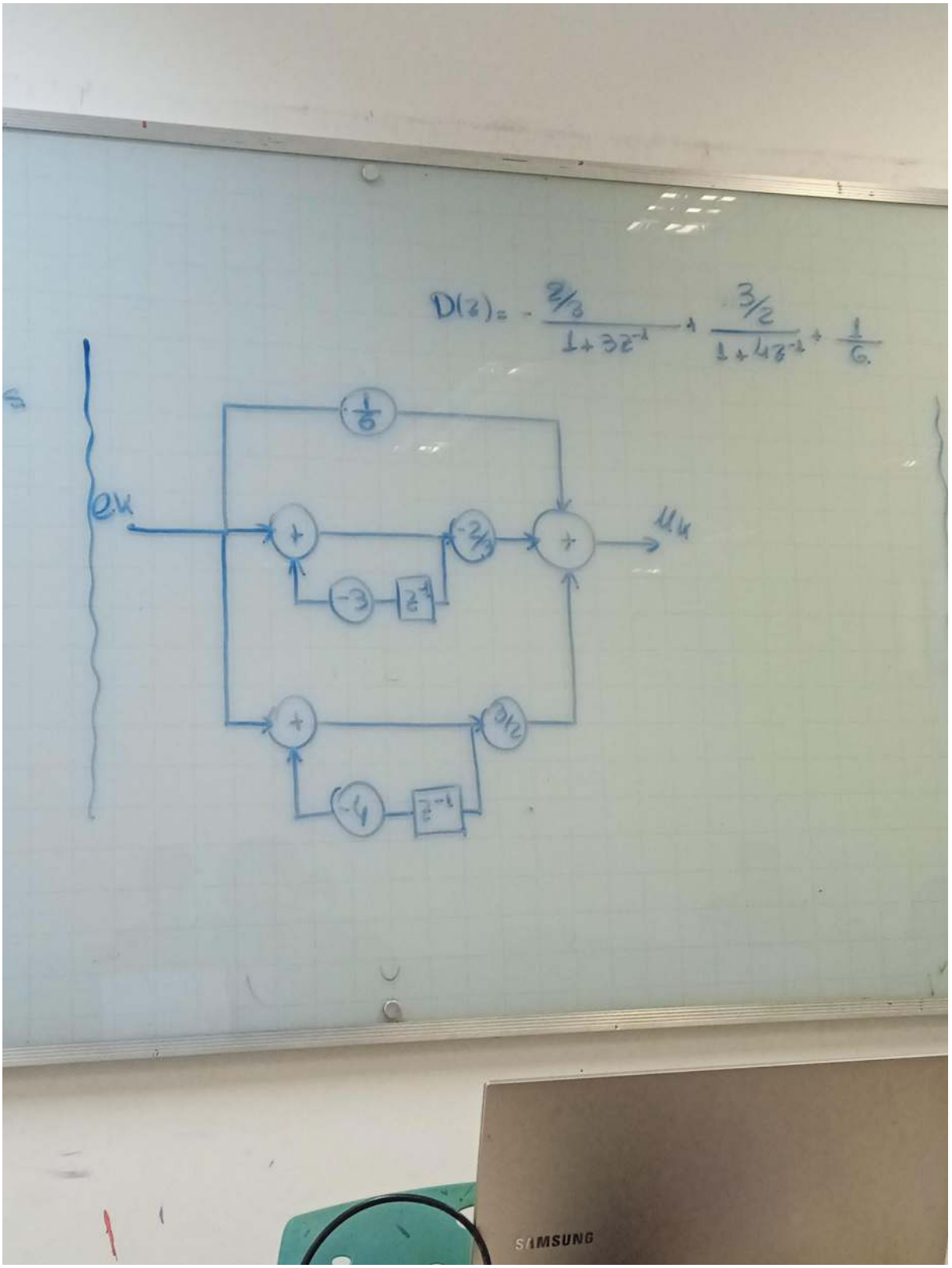
Estrutura Paralela

Continuando o exercício da aula passada (juntar)

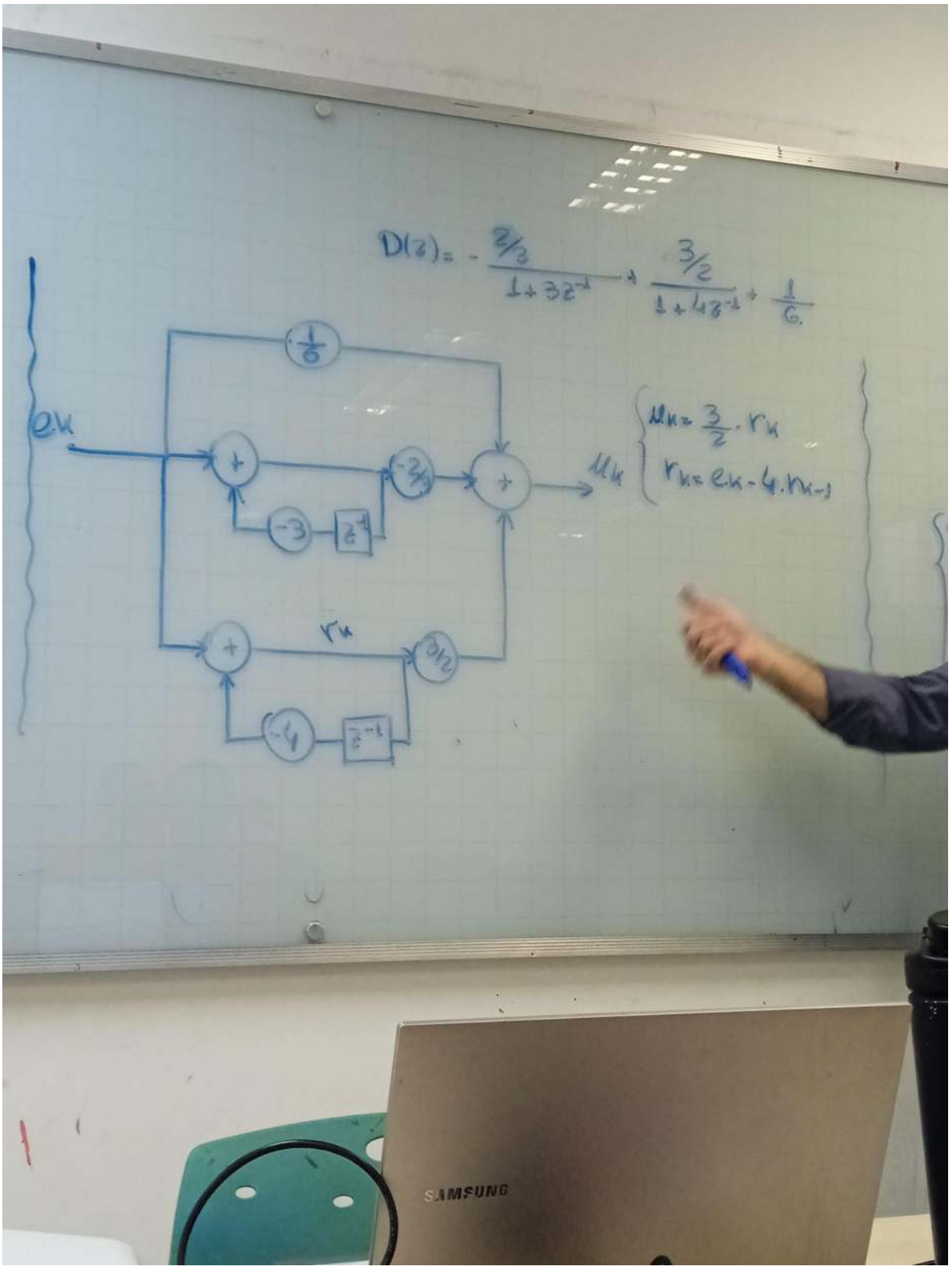


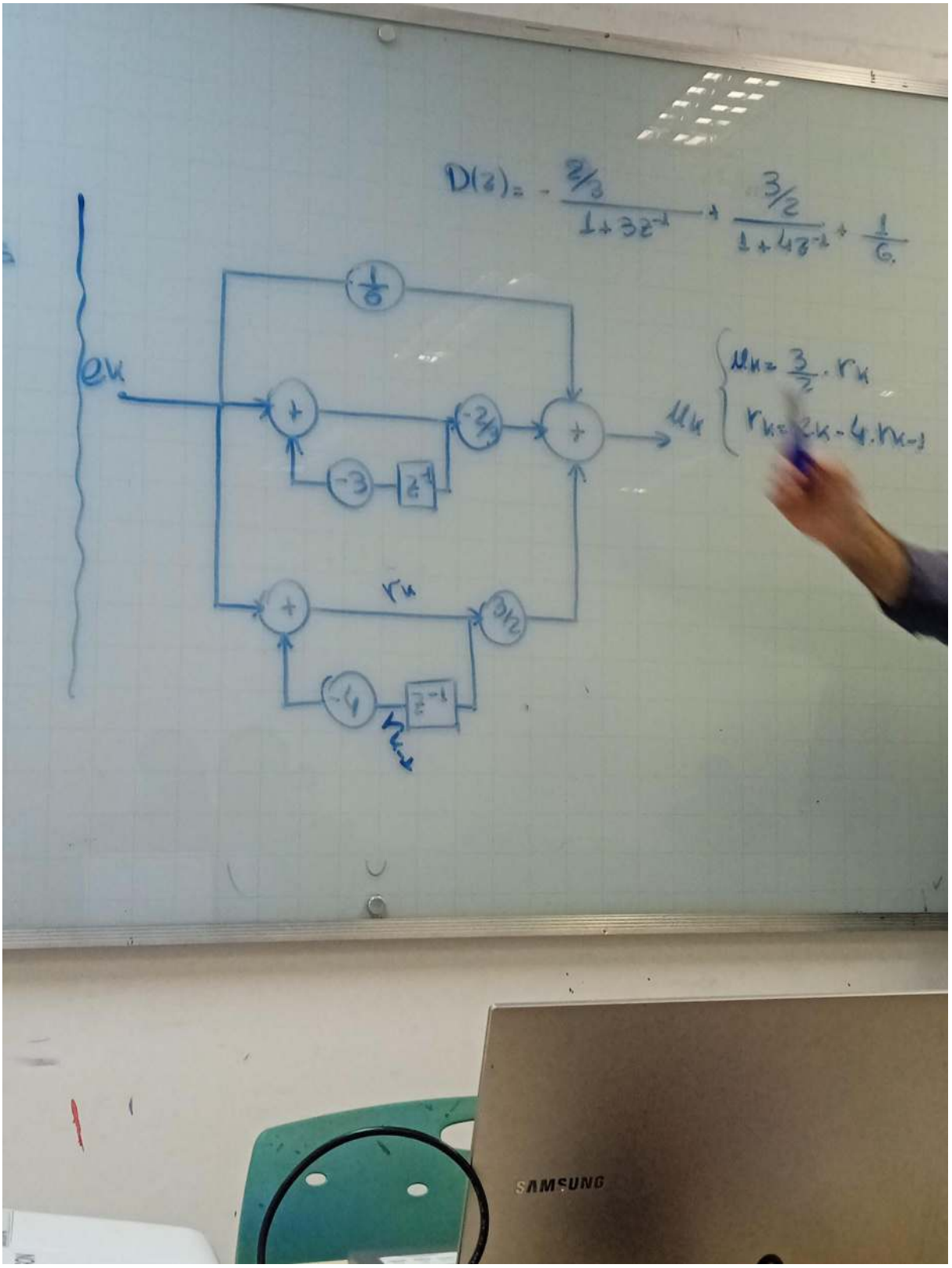
Resolução

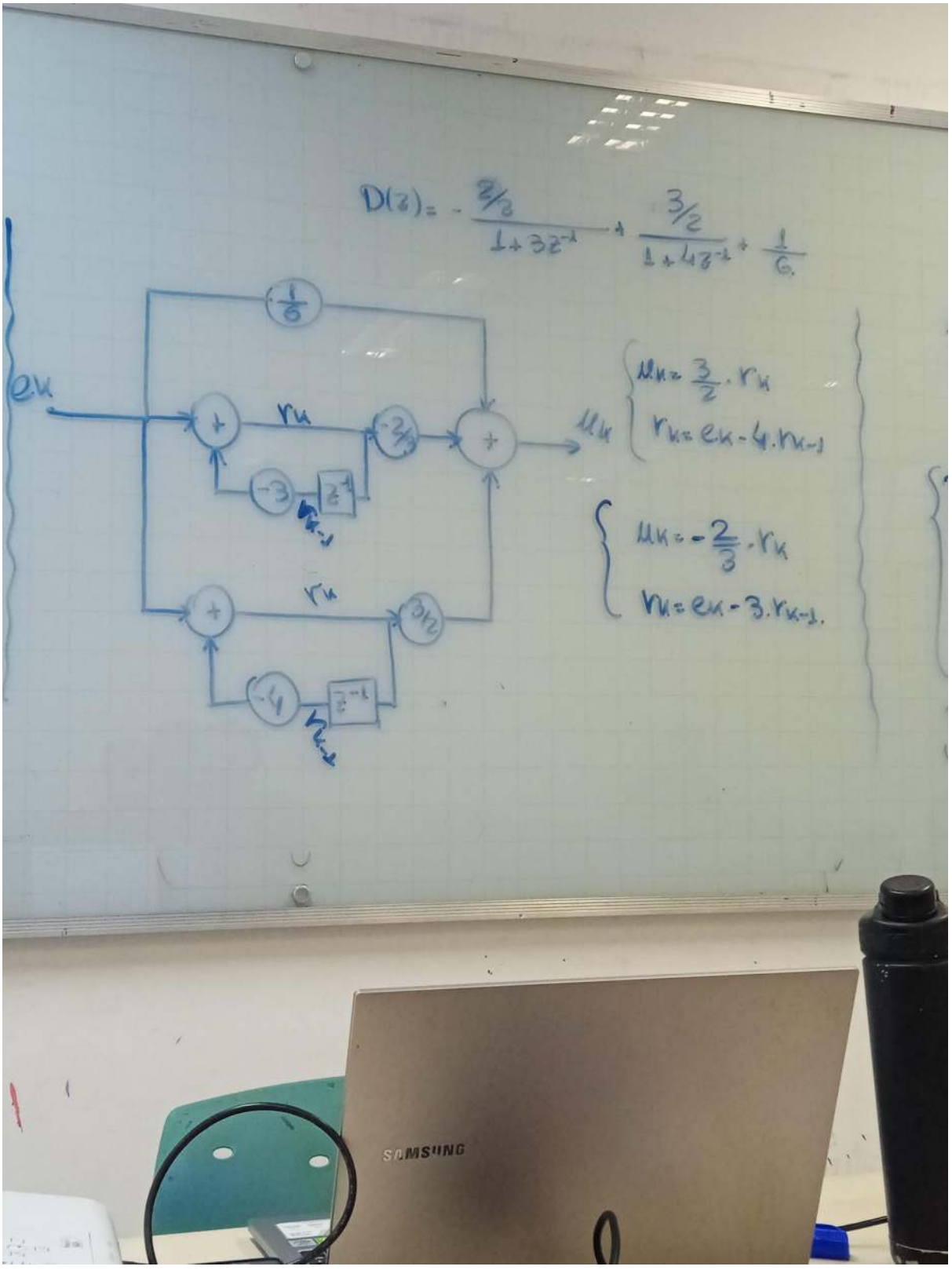
Usando a estrutura direta Canônica



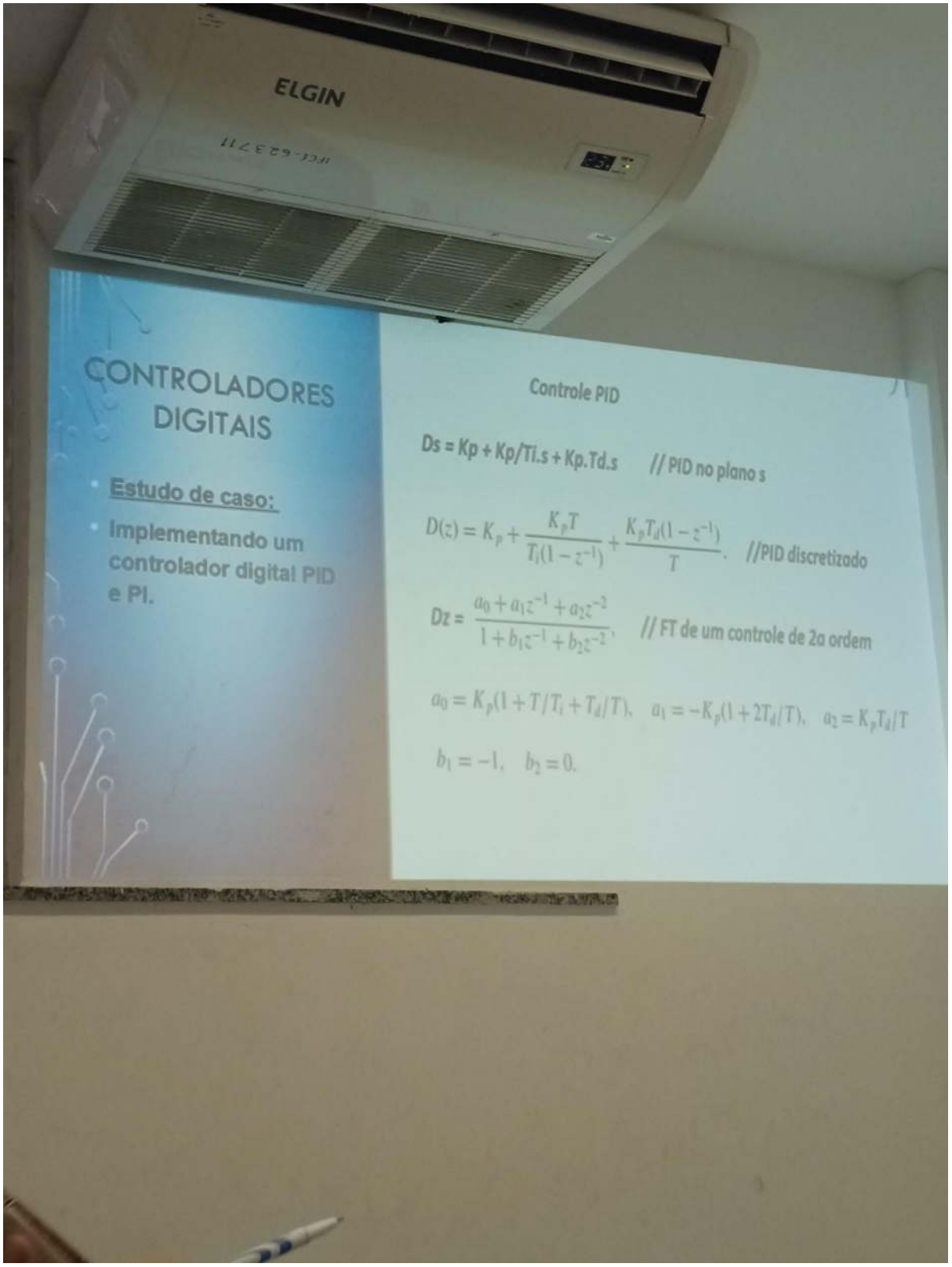
Encontrando uk e rk



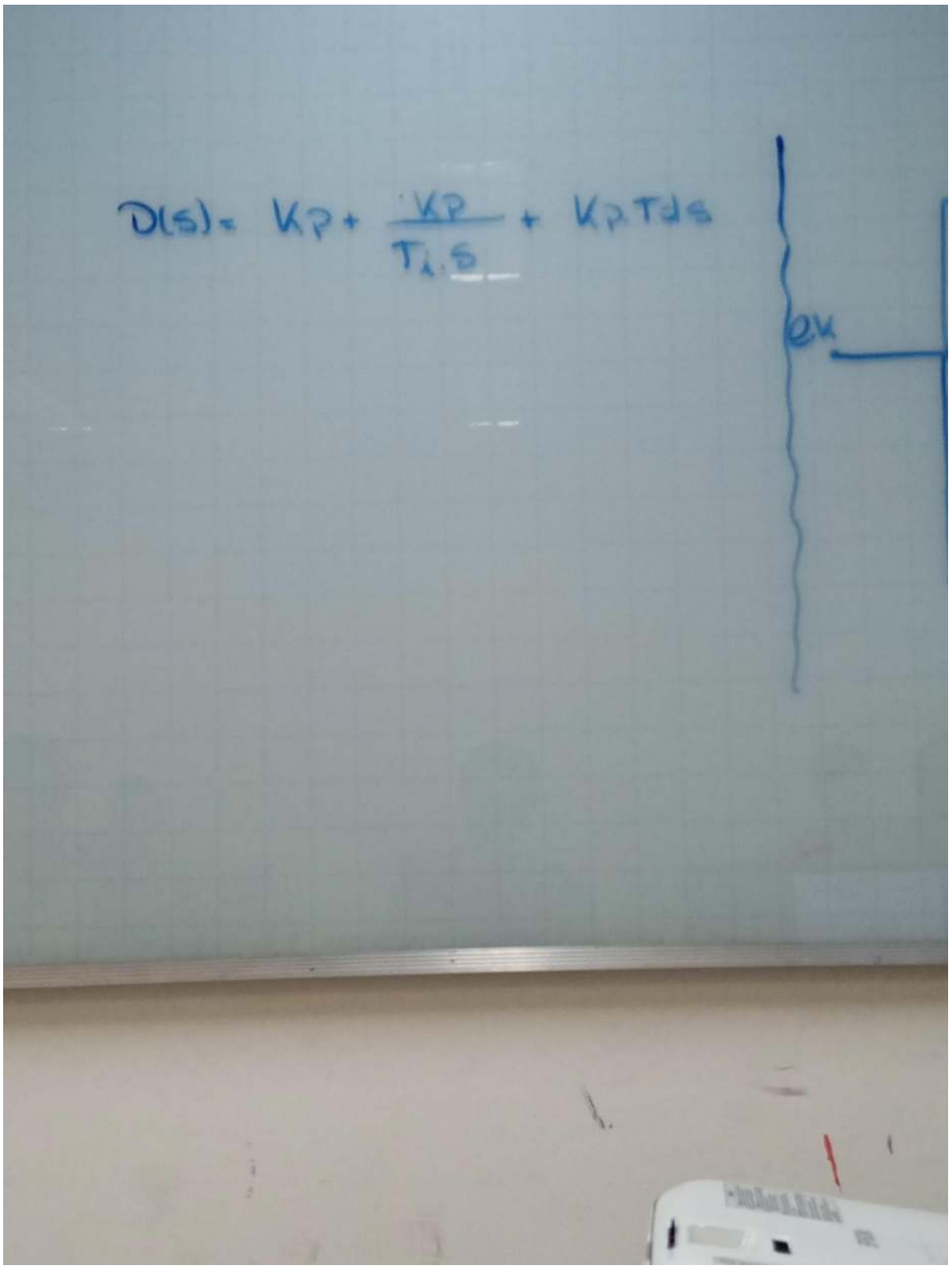

$D(z) = -\frac{2/3}{1+3z^{-1}} + \frac{3/2}{1+4z^{-1}} + \frac{1}{6}$
e_k
$-\frac{1}{6}$
$-\frac{2}{3}$
-3
z^{-1}
u_k
$u_k = \frac{3}{2} \cdot r_k$
$r_k = e_k - 4 \cdot r_{k-1}$
r_k
$\frac{3}{2}$
-4
z^{-1}
SAMSUNG



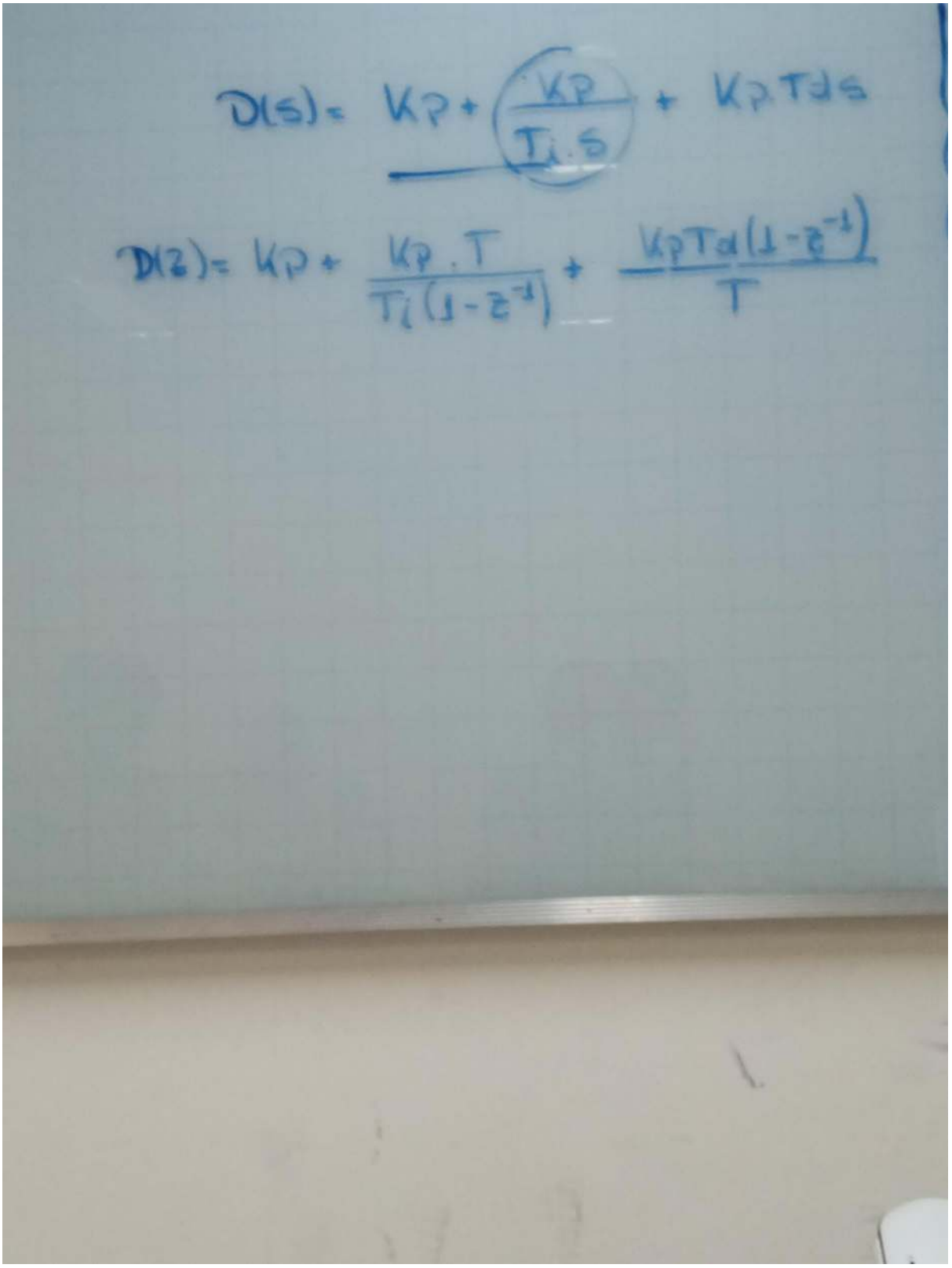
Controlador PID



Controlador PID discretizado no plano S

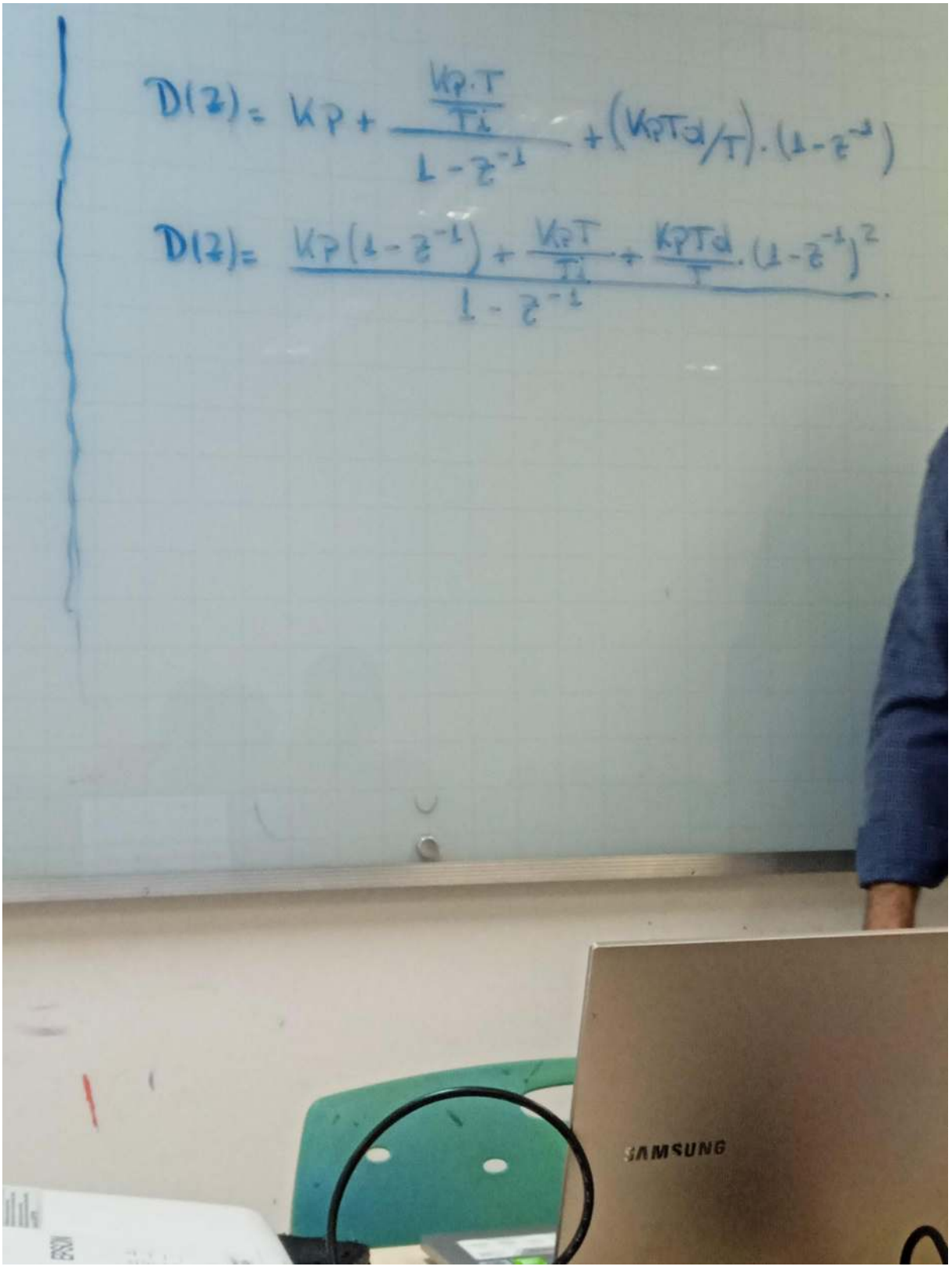


Discretizando no plano Z

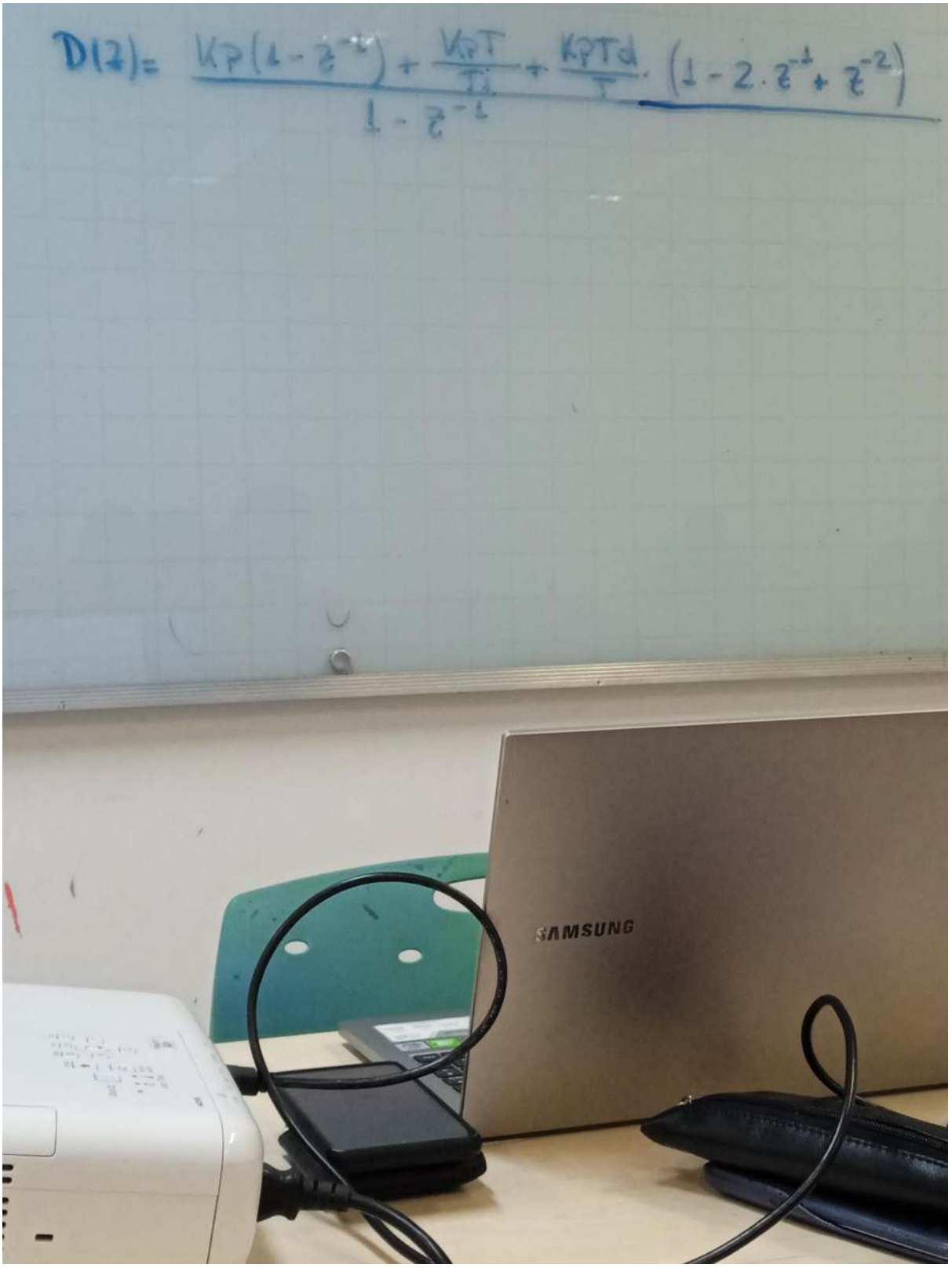


Kp: Ganho na parte proporcional
T: Tempo de amostragem
Ti: Tempo de ação da parte integral
Td: Tempo de ação da parte Derivativa

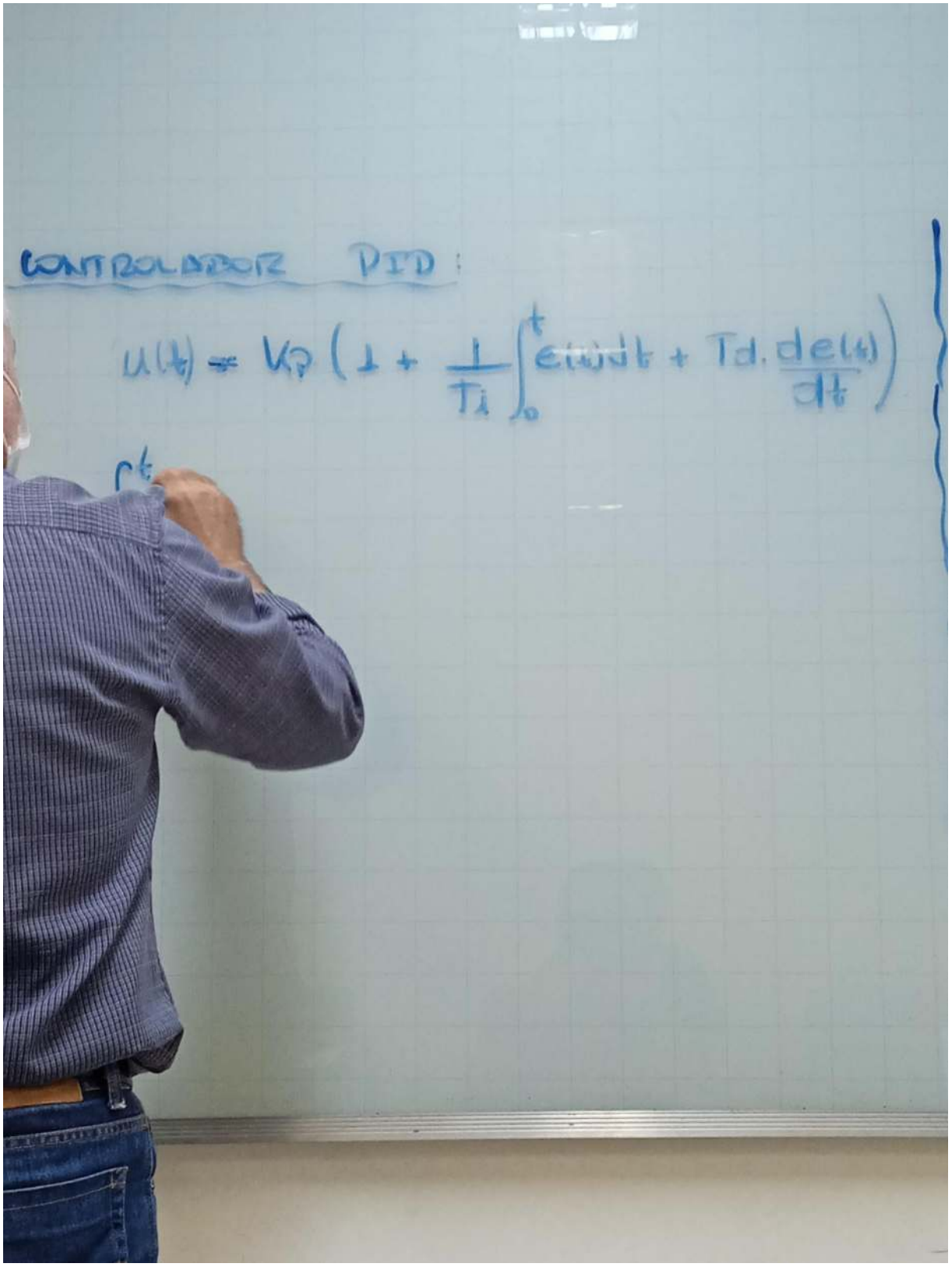
Resolução



Reescrevendo (1- z^-1)^2



Depois de resolver o numerador, será possível encontrar a0, a1, a2 e b1



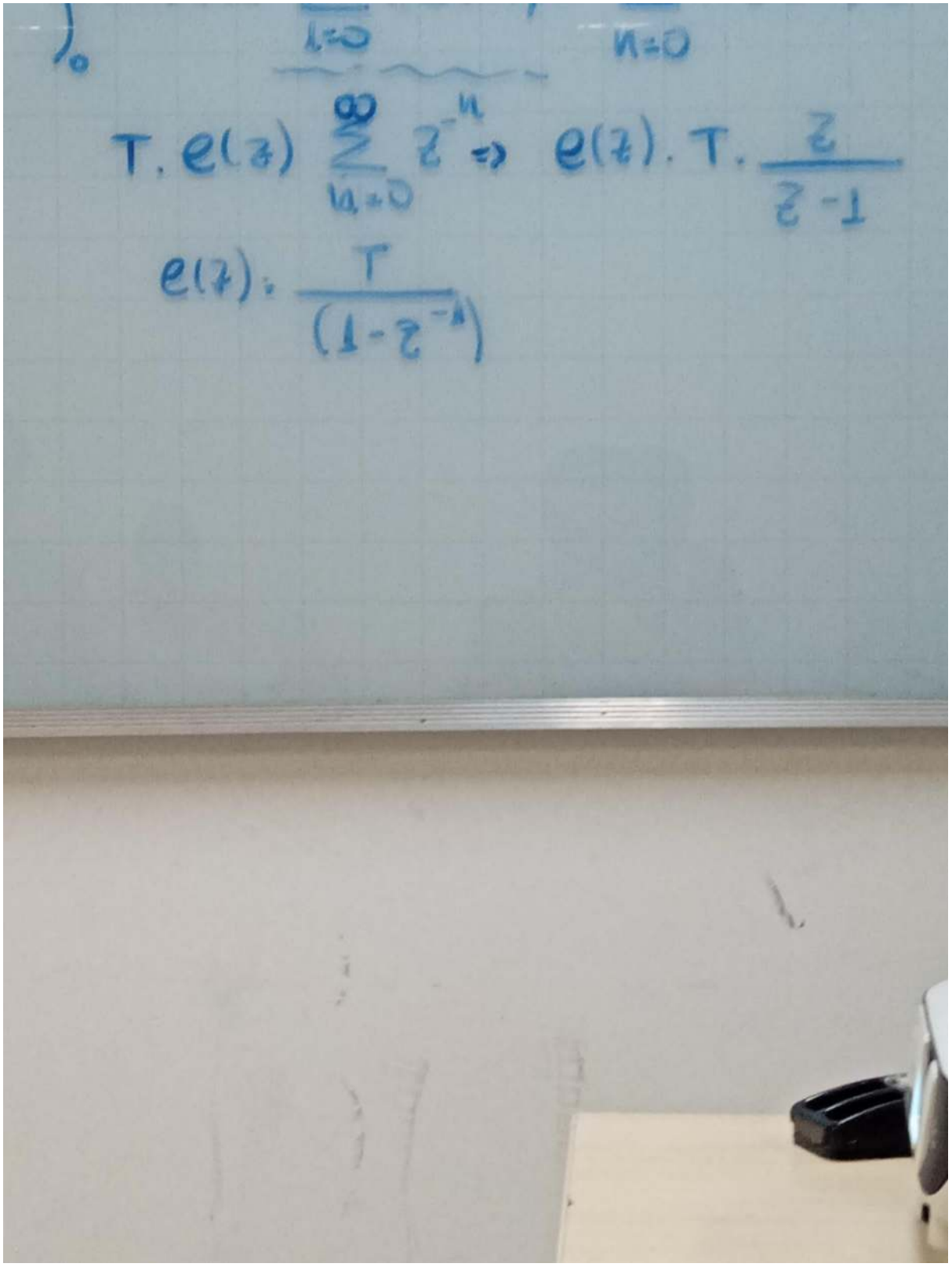
Fazendo uma transformação trapezoidal e representando no plano Z

CONTROLADOR PID:

$$u(t) = K_p\left(1 + \frac{1}{T_i}\int_0^t e(t)dt + T_d\cdot\frac{de(t)}{dt}\right)$$

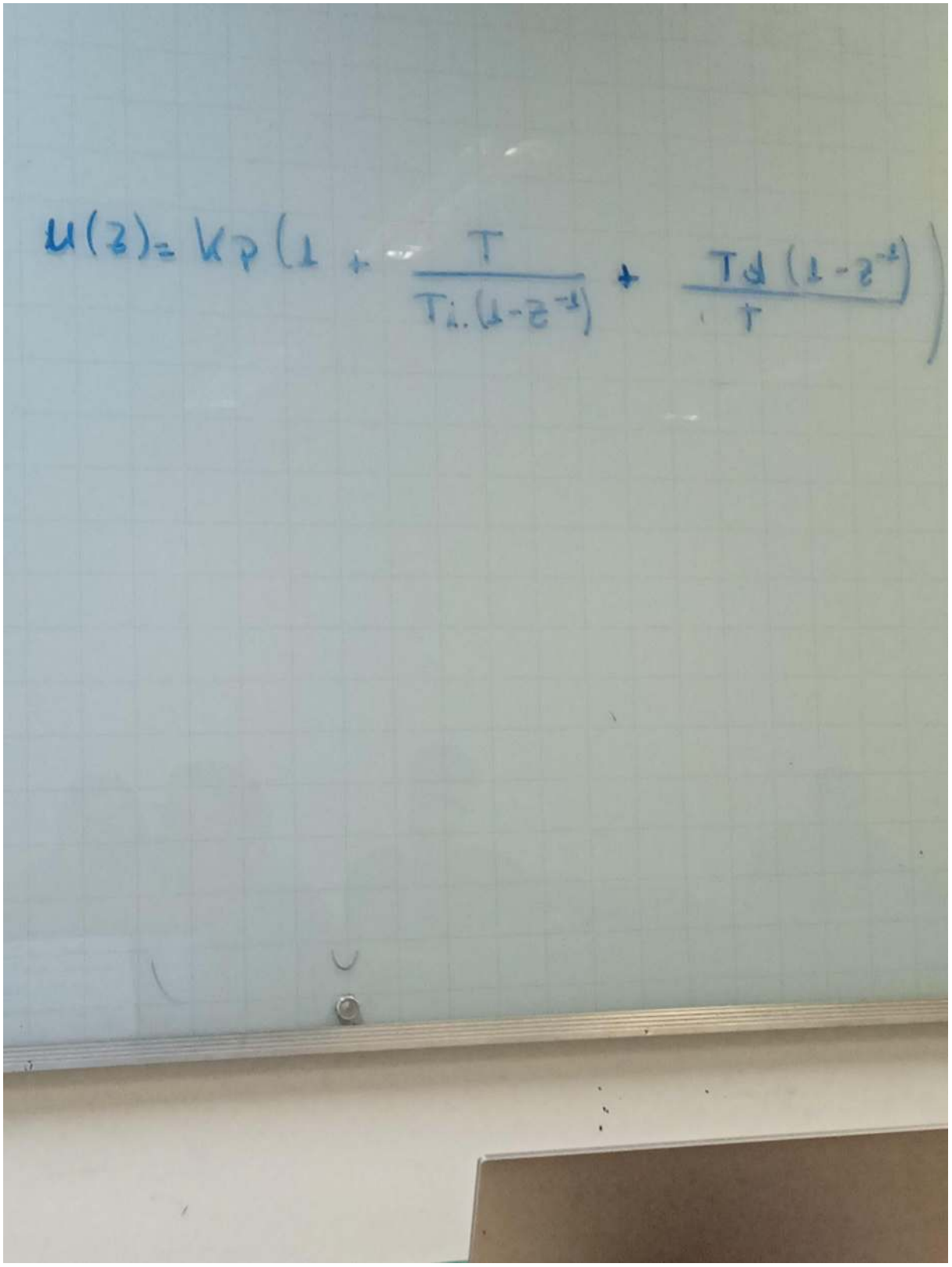
$$\int_0^t e(t)dt \approx \sum_{k=0}^{n} T\,e(kT) = \sum_{n=0}^{\infty} T\cdot e(kT)\cdot z^{-n}$$

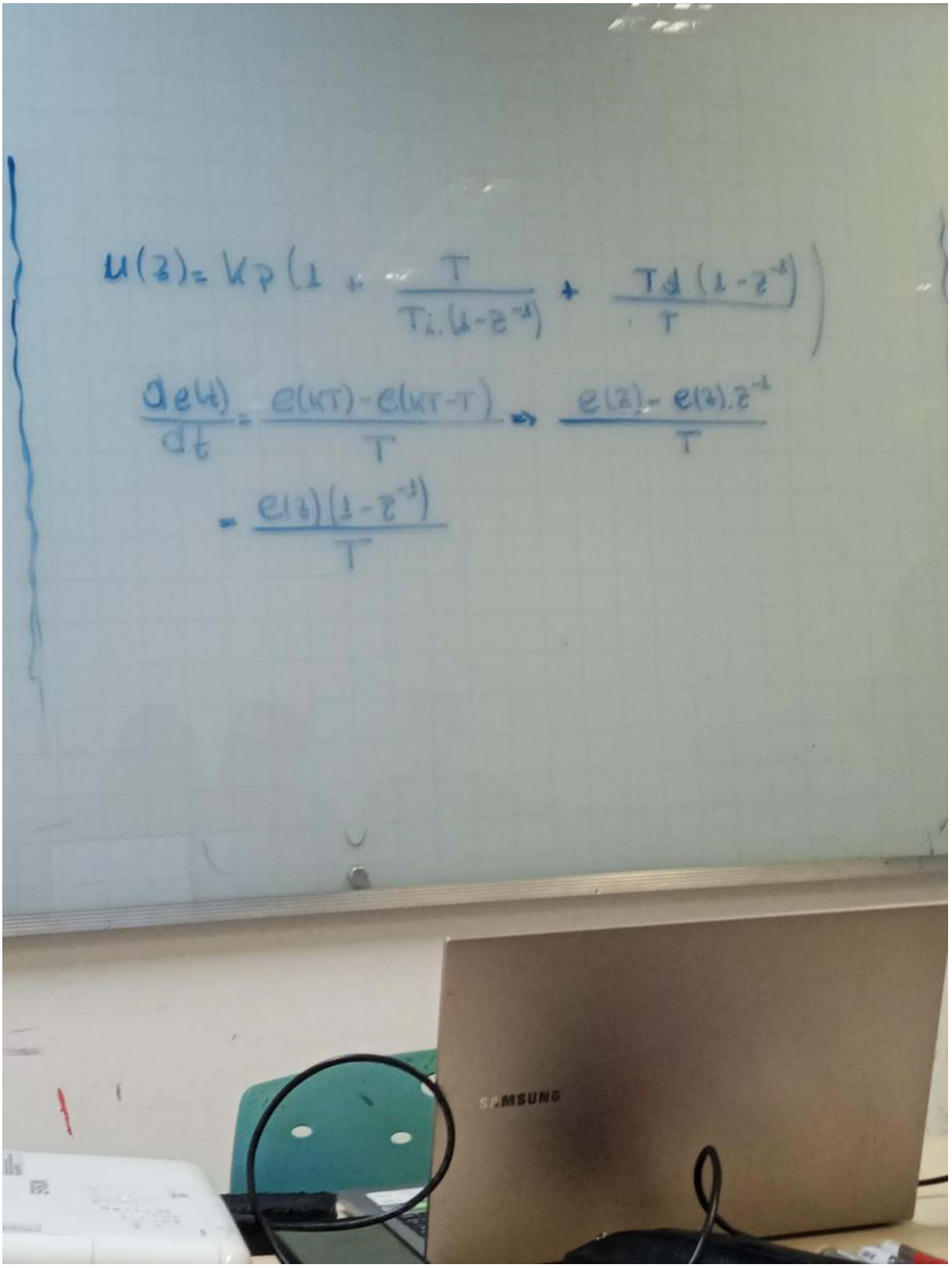
$$T\cdot E(z)\sum_{k=0}^{\infty} z^{-n}$$



O somatório é igual a:
Z/(Z-1)

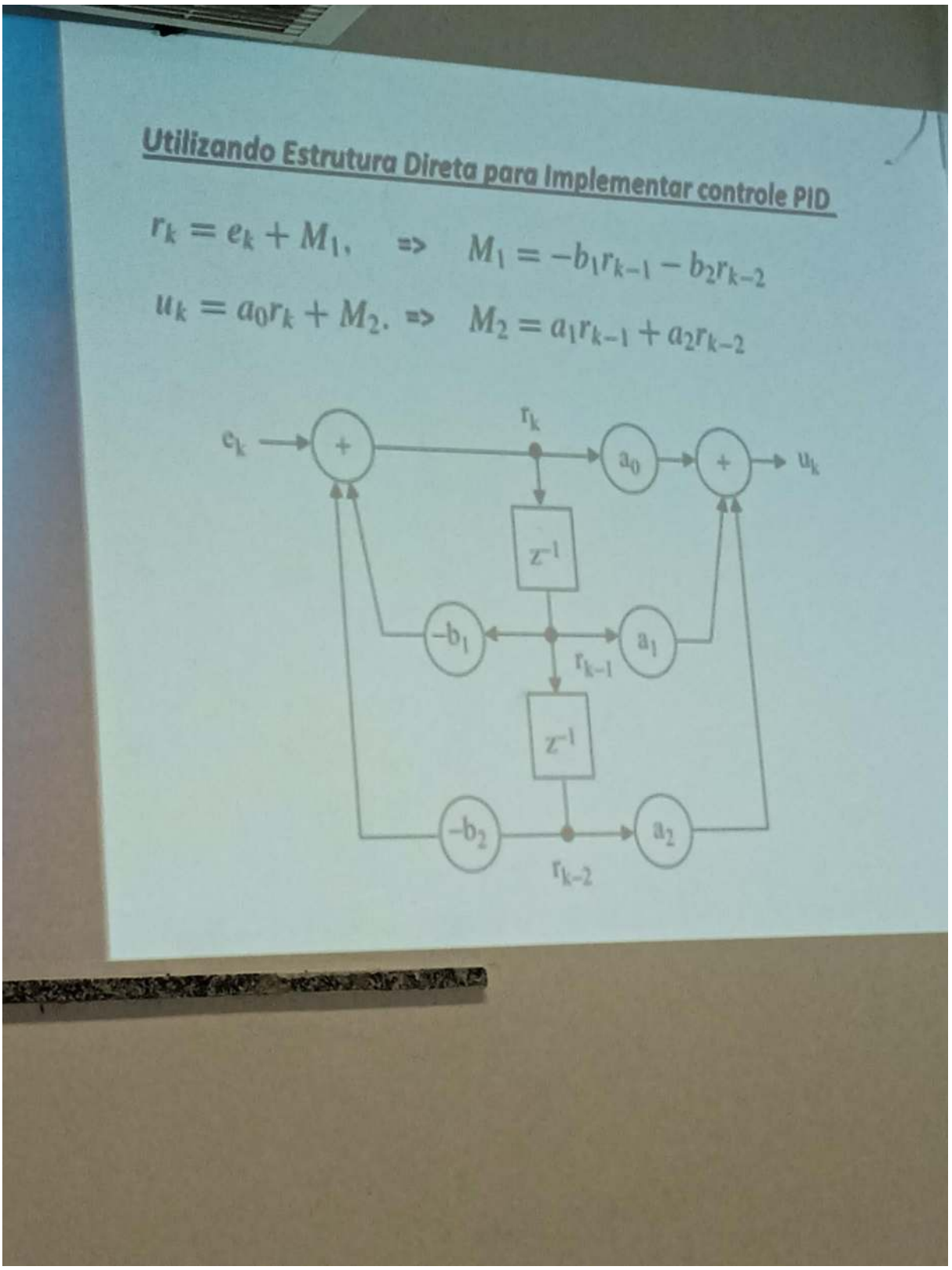
Encontrando a parte derivativa


$U(z) = K_p \left(1 + \frac{T}{T_i \cdot (z-1)} + \frac{T_d \cdot (z-1)}{T}\right)$

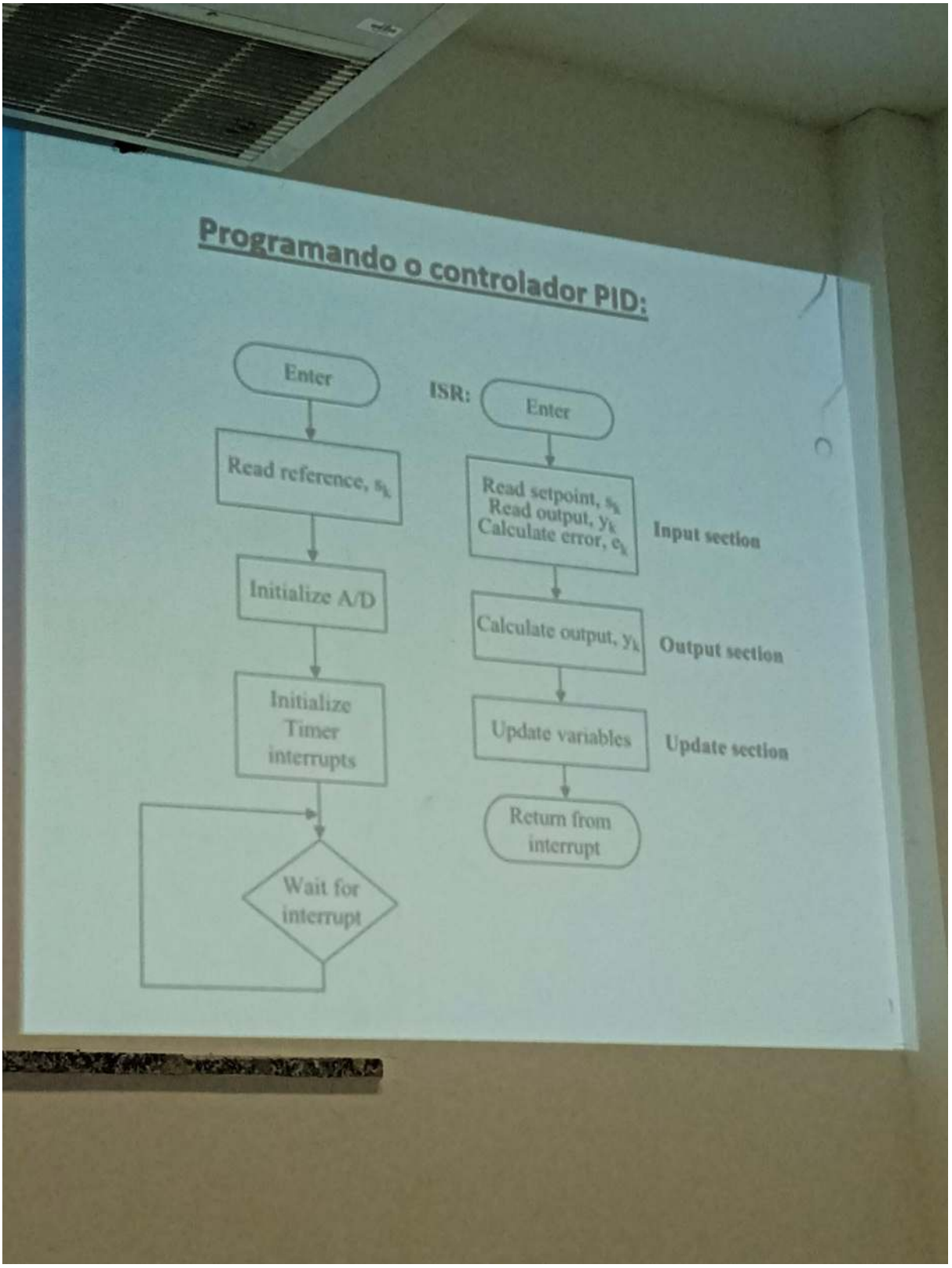


Dica: evitem decorar a expressão, tentem chegar na expressão a partir de Uk e Rk

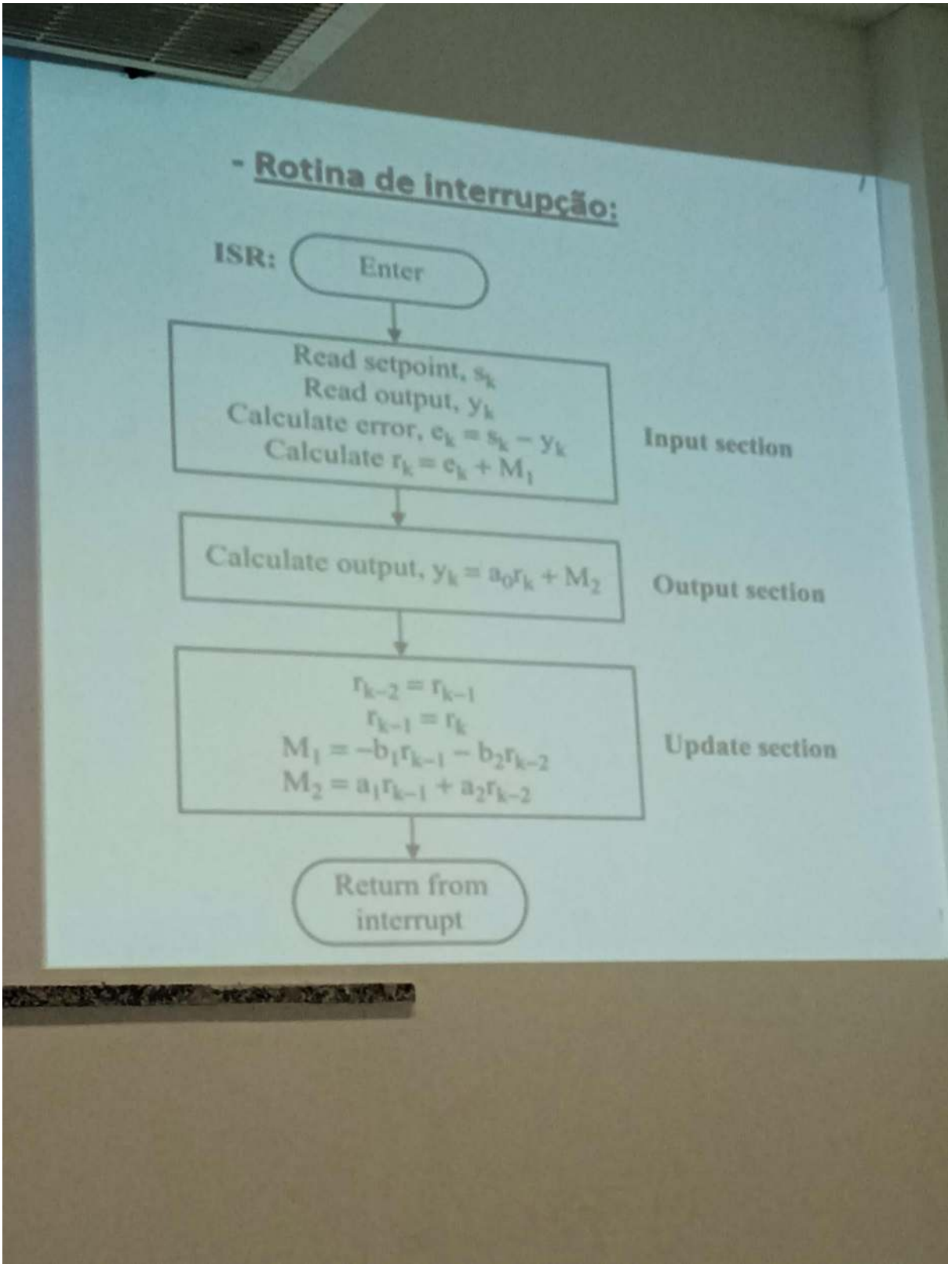
Montando as equações de blocos



Programando o Controlador PID



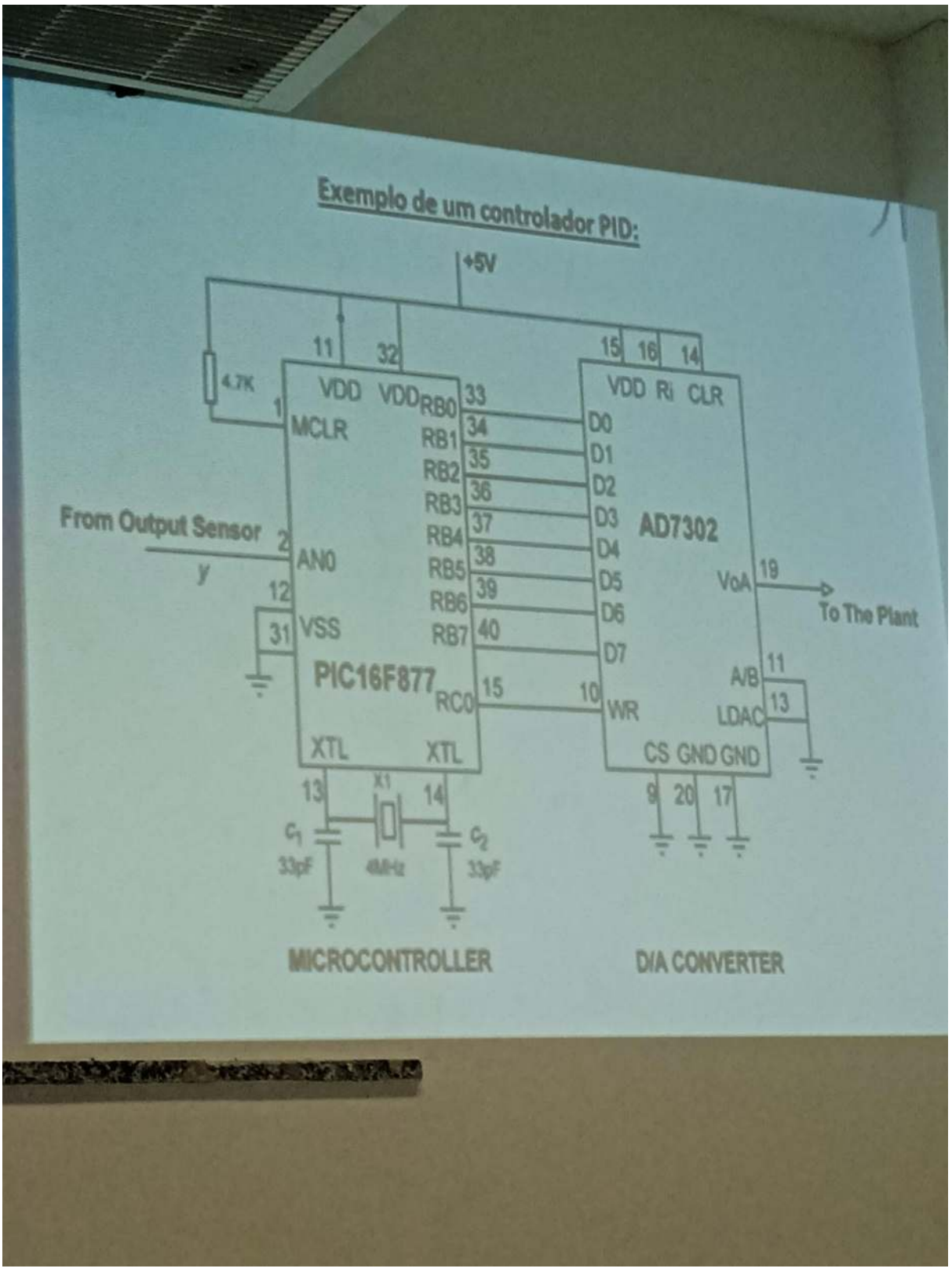
Continuando: Rotina de interrupção



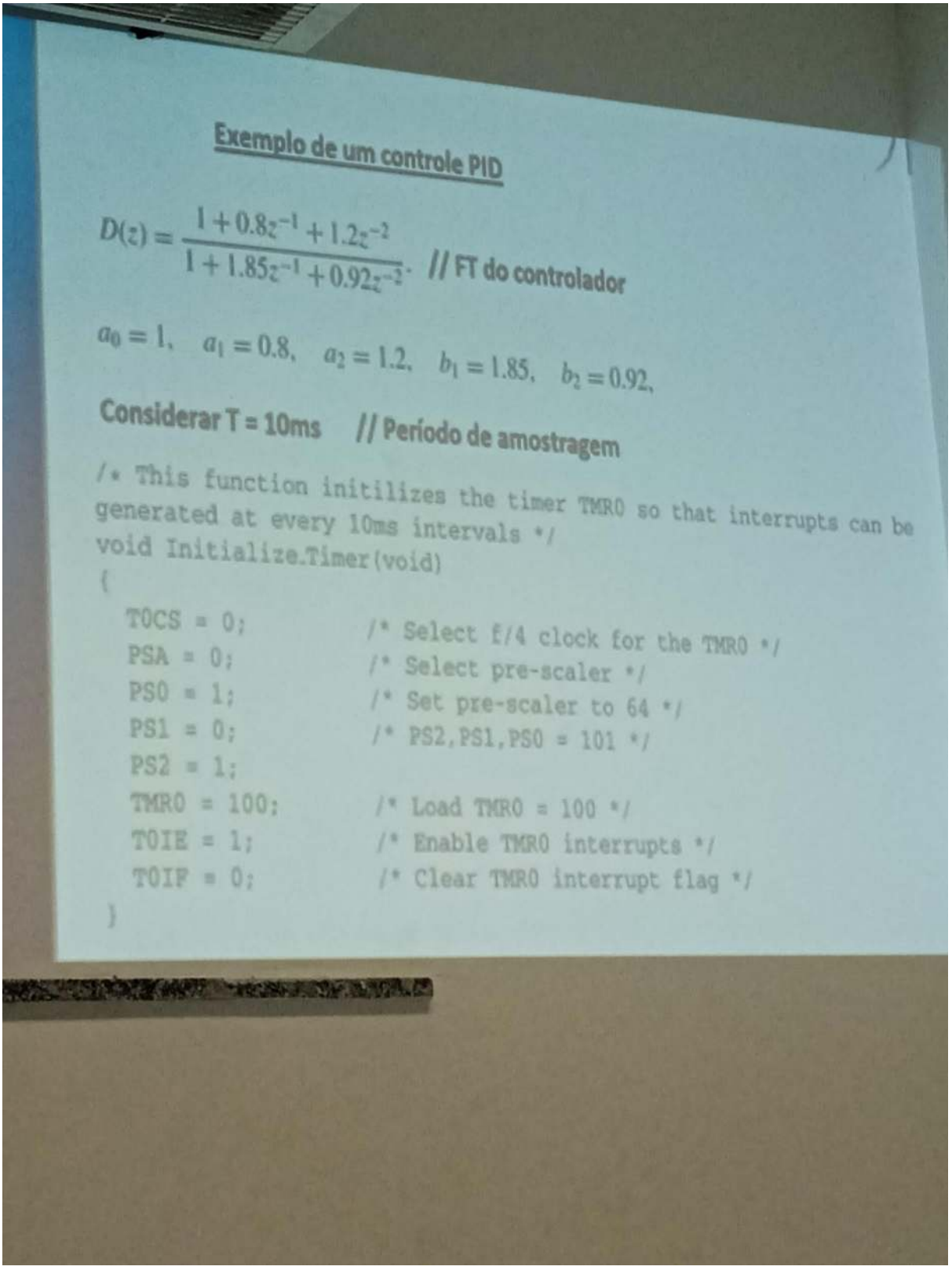
Rk pode ser usado o cálculo do quadro, se fizer isso o M1 não será necessário

Outra forma: Se não usar o interrupção do timer, insira um atraso

Exemplo de um controlador PiD



Código em linguagem em C



Exemplo de um controle PID

$$D(z) = \frac{1 + 0.8z^{-1} + 1.2z^{-2}}{1 + 1.85z^{-1} + 0.92z^{-2}}.$$ // FT do controlador

$a_0 = 1, \quad a_1 = 0.8, \quad a_2 = 1.2, \quad b_1 = 1.85, \quad b_2 = 0.92,$

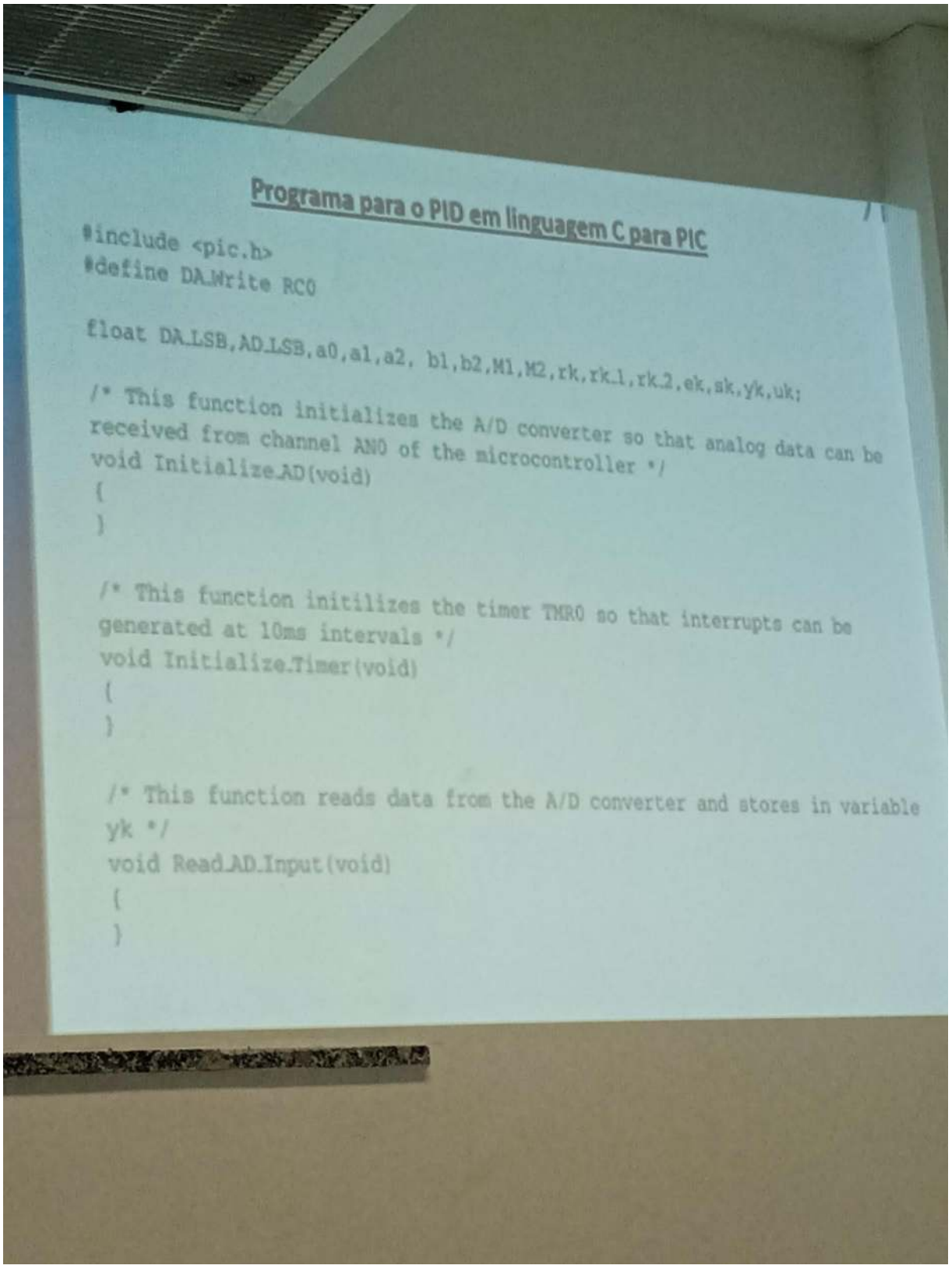
Considerar T = 10ms // Período de amostragem

```
/* This function initilizes the timer TMR0 so that interrupts can be
generated at every 10ms intervals */
void Initialize.Timer(void)
{
  T0CS = 0;          /* Select f/4 clock for the TMR0 */
  PSA = 0;           /* Select pre-scaler */
  PS0 = 1;           /* Set pre-scaler to 64 */
  PS1 = 0;           /* PS2,PS1,PS0 = 101 */
  PS2 = 1;
  TMR0 = 100;        /* Load TMR0 = 100 */
  T0IE = 1;          /* Enable TMR0 interrupts */
  T0IF = 0;          /* Clear TMR0 interrupt flag */
}
```

A frequência de clock interna(fcint) é 1MHz

frequência de incremento do timer interno (fT) é 1/64 = 15625Hz ou 1/fT = 64us

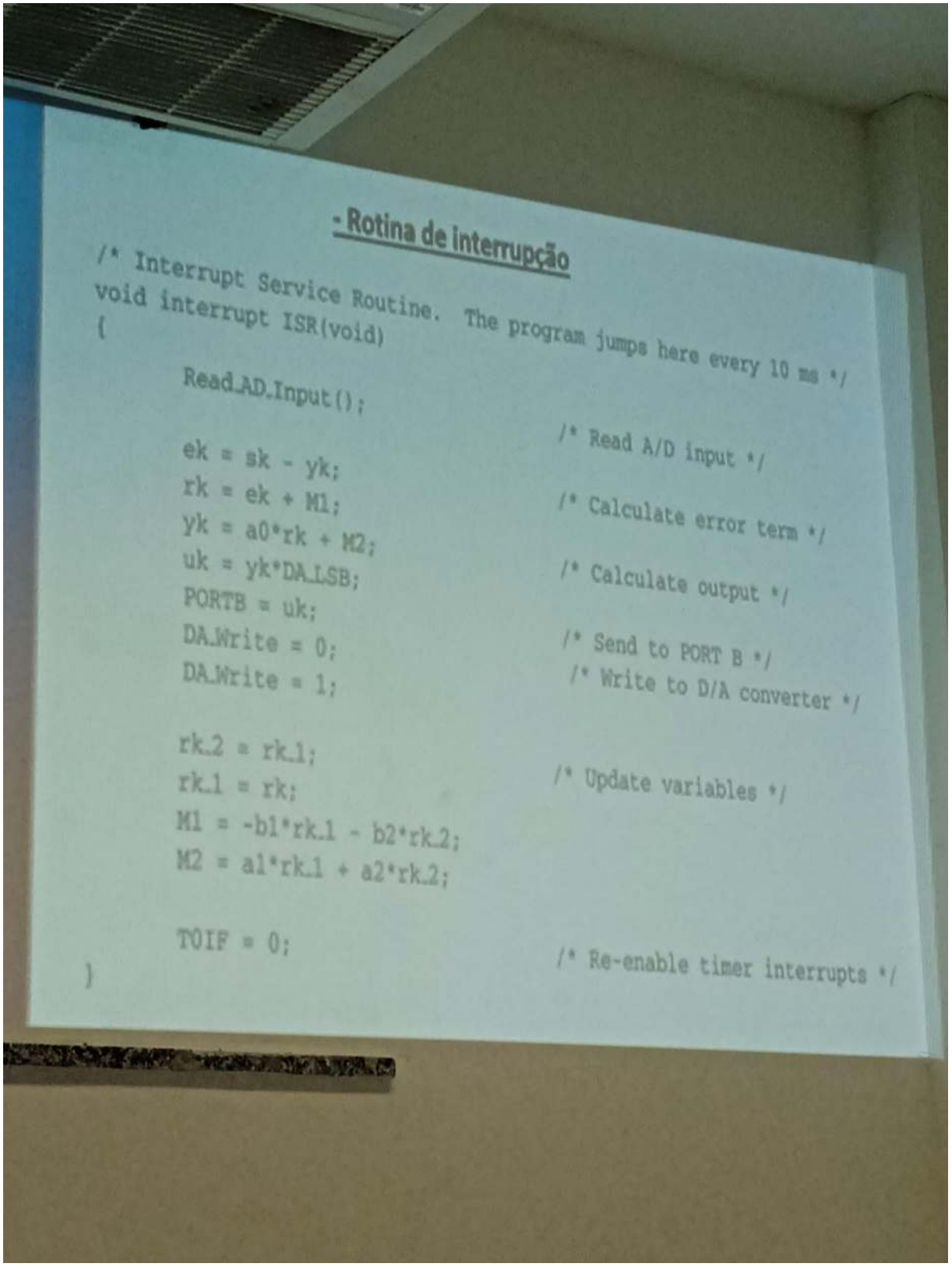
REG = 8bits = 2^8 = 256 -100 = 156 x 64us ≈10 ms



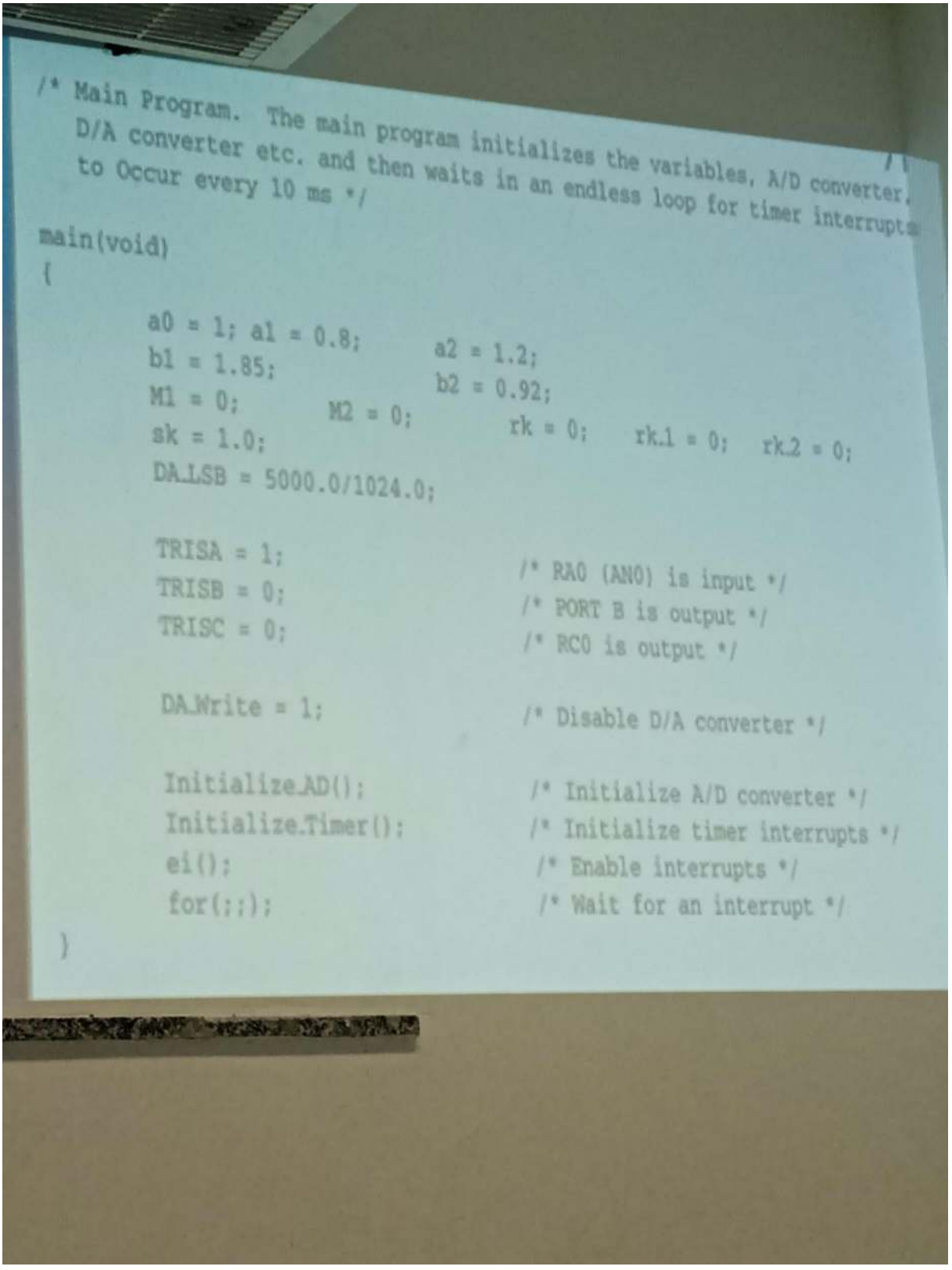
Programa para o PID em linguagem C para PIC

```
#include <pic.h>
#define DA_Write RC0

float DA_LSB,AD_LSB,a0,a1,a2, b1,b2,M1,M2,rk,rk_1,rk_2,ek,sk,yk,uk;

/* This function initializes the A/D converter so that analog data can be
received from channel AN0 of the microcontroller */
void Initialize_AD(void)
{
}

/* This function initilizes the timer TMR0 so that interrupts can be
generated at 10ms intervals */
void Initialize_Timer(void)
{
}

/* This function reads data from the A/D converter and stores in variable
yk */
void Read_AD_Input(void)
{
}
```

Rotina de interrupção



A rotina de interrupção é chamado a cada 10ms



```
/* Main Program.  The main program initializes the variables, A/D converter,
   D/A converter etc. and then waits in an endless loop for timer interrupts
   to Occur every 10 ms */

main(void)
{
        a0 = 1; a1 = 0.8;       a2 = 1.2;
        b1 = 1.85;              b2 = 0.92;
        M1 = 0;         M2 = 0;         rk = 0;    rk_1 = 0;   rk_2 = 0;
        sk = 1.0;
        DA_LSB = 5000.0/1024.0;

        TRISA = 1;                      /* RA0 (AN0) is input */
        TRISB = 0;                      /* PORT B is output */
        TRISC = 0;                      /* RC0 is output */

        DA_Write = 1;                   /* Disable D/A converter */

        Initialize_AD();                /* Initialize A/D converter */
        Initialize_Timer();             /* Initialize timer interrupts */
        ei();                           /* Enable interrupts */
        for(;;);                        /* Wait for an interrupt */
}
```

DA.LSB é para converter a informação decimal e Milivolts para Volt

Todos os pinos da porta A são de entrada ao atribuir 1

Todos os pinos da porta B e C são de saída ao atribuir 0

Funções para inicializar e ler o conversor AD:

```
/* This function initializes the A/D converter so that analog data
can be received from channel AN0 of the microcontroller */
void Initialize_AD(void)
{
  ADCON1 = 0 x 8E;          /* Configure AN0 for +5V reference */
  ADCON0 = 0 x 41;          /* Select A/D converter clock */
}
/* This function reads data from the A/D converter and stores in
variable yk */
void Read_AD_Input(void)

{
ADCON0 = 0 x 45;                /* Start A/D conversion */
while(ADCON0 & 4) != 0);        /* Wait until conversion completes */
y_high = ADRESH;                /* High 2 bytes of converted data */
y_low = ADRESL;                 /* Low byte of converted data */
yk = 256.0*y_high + y_low;      /* Converted data in yk */
yk = yk*AD_LSB;                 /* Sensor output in mV */
}
```

AD_LSB converts the A/D value to into millivolts.

Quinta-feira

- 90% do projeto desde o zero até fazer o programa em C